NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMAGE SEGMENTATION FOR IMPROVISED EXPLOSIVE DEVICES

by

Danny Heerlein

December 2012

Thesis Advisor:	Nedialko B. Dimitrov
Second Reader:	Jason C. Caldwell

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
3–12–2012	Master's Thesis	2011-10-01—2012-11-30

4. TITLE AND SUBTITLE

Image Segmentation for Improvised Explosive Devices

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Danny Heerlein

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

N/A

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A

14. ABSTRACT

This thesis creates algorithms to preprocess colored images in order to segment Improvised Explosive Devices (IEDs). IEDs are usually concealed and camouflaged and therefore more difficult to segment than other objects. We address the increased difficulty with three key contributions: 1) Our algorithm automatically divides a user-defined background area into smaller areas. We generate separate color models for each of these areas to ensure that a color model includes only colors that appear in the same area of the background. 2) We compress each of these complex color models into a statistical model. This increases the number of background models we can hold simultaneously in working memory, and allows us to generate a set of background models that describes a complete environment. 3) We estimate the initial object labels based on the color distance to the background. This approach allows us to generate color models for IEDs without user input that labels parts of the IED.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	89	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

IMAGE SEGMENTATION FOR IMPROVISED EXPLOSIVE DEVICES

Danny Heerlein
Captain, German ARMY
M.S. (Diplom-Informatiker), University of the German Federal Armed Forces Munich

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATION RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
December 2012

Author: Danny Heerlein

Approved by: Nedialko B. Dimitrov
Thesis Advisor

Jason C. Caldwell
Second Reader

Robert F. Dell, Ph.D.
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis creates algorithms to preprocess colored images in order to segment Improvised Explosive Devices (IEDs). IEDs are usually concealed and camouflaged and therefore more difficult to segment than other objects. We address the increased difficulty with three key contributions: 1) Our algorithm automatically divides a user-defined background area into smaller areas. We generate separate color models for each of these areas to ensure that a color model includes only colors that appear in the same area of the background. 2) We compress each of these complex color models into a statistical model. This increases the number of background models we can hold simultaneously in working memory, and allows us to generate a set of background models that describes a complete environment. 3) We estimate the initial object labels based on the color distance to the background. This approach allows us to generate color models for IEDs without user input that labels parts of the IED.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Improvised Explosive Devices (IEDs), in different forms, are a favorite weapon for global insurgents and terrorists, with an average of 260 IED incidents per month in Afghanistan and Iraq for the time from January 2004 until May 2010. Terrorist's ability to quickly change the IED type requires a flexible approach in IED detection training. To create an effective training tool, an image taken in the field must be segmented into two sections, separating the IED and the image background.

Different from objects typically used to evaluate the quality of segmentation algorithms, IEDs are not easy to detect even for humans. They are purposefully hard to detect, concealed or colored very similar to the background. This thesis evaluates existing algorithms for image segmentation and creates new algorithms to pre-process colored images in order to segment IEDs.

We first modify a basic graph cut algorithm to segment completely visible IEDs from a uniformly colored background. To measure the performance depending on the parameter settings, we run a series of tests defined by a Nearly Orthogonal Latin Hypercube design of experiment.

A second series of tests shows that the basic approach does not result in good segmentations if applied to realistic IEDs. We identify two major issues that are due the special nature if IEDs:

- Different from other segmentation tasks, real-world IEDs are only a small fraction of the complete image. This results in a training area for the background model that is extremely large compare to the training area of the IED.
- IEDs do not necessary appear as a single object. Often, charges and initiation systems appear -if visible- as different objects. Visible parts of an IEDs can be small and partially hidden. Therefore, a single, manually provided training area of an IED is not precise enough.

We address the increased difficulty while segmenting IEDs with three key contributions:

1. Our algorithm automatically divides a user-defined background area into smaller areas. We generate separate color models for each of these areas, to ensures that a color model includes only colors that appear close together in the background. For each of these areas, we build a color model that represents the complete RGB color space of the area.
2. We compress each of these complex color models into a statistical model by applying the k-means Clustering algorithm for up to 10 clusters. This compressed statistical description increases the number of background models we can hold simultaneously in working memory.

3. We use the estimated distance to the background to estimate the initial object label automatically. In this way, we can segment an IED without user input labeling the IED.

The results of this thesis have three different applications: 1) defining the IED on an image, 2) generating new training images, 3) helping humans detect IEDs.

In order to distinguish between a student's detection and misdetection of an IED in a colored image, the area of an IED must be defined. Our work defines this area through the set of pixels at the end of the graph cut algorithm.

To generate new training images, a segmented IED can be combined with images showing a different background. Our work facilitates this through precise IED segmentation.

Finally, the automated IED labels we generate may be helpful as a hybrid between a computer pre-processing the image, and a human operator providing the final identification. The automatic label computation is efficient and can be computed in real-time.

Acknowledgements

I would like to express the deepest appreciation to my thesis advisor, Professor Nedialko B. Dimitrov. He continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this thesis would not have been possible.

In addition, a thank-you to LTC Jason Caldwell, Simulation Officer and Combat Analyst at the Training and Doctrine Command —Monterey, whose support and enthusiasm had lasting effect and raised many ideas regarding possible future applications.

This thesis is dedicated to my family, especially to my wife and my daughter, who have always stood by me and dealt with all of my absences from many family occasions with a smile.

Last, but not least, I am indebted to my many class-mates who supported my work by providing feedback and ideas to improve the application we developed.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
INTRODUCTION

1.1 Motivation of Our Study

1.1.1 Background and Area of Research

Improvised Explosive Devices (IEDs), in different forms, are a favorite weapon for global insurgents and terrorists, with an average of 260 IED incidents per month in Afghanistan and Iraq. The use of IEDs has become a generic global threat [1], [2]. IEDs consist mainly of charges and initiation systems. The charges are often available from past armed conflicts, and the initiation systems are often commercial detonators, readily available for legitimate purposes and often mass-produced [2]. Therefore, cheap and easy to make, IEDs have become the weapon of choice in unconventional warfare [3].

IED detection has two forms: the automatic detection with specialized equipment and manual detection by trained individuals. These two methods of detection are noncompetitive, and both serve to decrease the threat caused by IEDs.

Terrorists' ability to quickly change the IED type requires a flexible approach in IED detection training. The Training and Doctrine Command (TRADOC) Analysis Center-Monterey was asked to support Future Warrior Technology Integration (FWTI) by analyzing and designing a flexible application in the training for IED detection. This application is called the *Perceptual Learning Trainer* (PLT).

Actual photographs of IEDs and their surroundings are required to generate images for IED detection training and to generate templates for automatic IED detection. To create an effective training tool, an image taken in the field must be segmented into two separate sections, identifying the IED and the image background. This thesis focuses on analyzing image segmentation algorithms to perform this task.

1.1.2 The Importance of Cutting IEDs from a Photograph

The following scenarios explain the utility of separating an IED from the background, given an original combined image as input.

1. An instructor wants to train individuals to detect IEDs in real-world photographs:
   A photograph of an IED and surroundings usually consists of three regions: the actual IED, an unusual region around the IED, and the background. To distinguish between a student's detection and misdetection, the instructor requires an automatic method of defining these regions.

2. An instructor wants to create new training images in advance:
   IED types used in Iraq also tend to be used in Afghanistan after a certain amount of time. Clay Wilson, Specialist in Technology and National Security Foreign Affairs, Defense, and Trade Division, said in the *CRS Report for Congress* [4] that: "*Observers have indicated that the Taliban forces in Afghanistan appear to have learned some IED techniques from the Iraqi insurgents.*" Therefore, combining IEDs detected in Iraq and a background from Afghanistan could be helpful in generating flexible training. One way of doing this is to create a knowledge base of IED types and their appearance. Since the new background could be different than the original background, the IED image has to be segmented from the background image so that there are as few as possible background pixels left on the IED image.

3. IED images are used to automatically identify if an IED is present in an image:
   General approaches for object recognition based on templates (*Pattern Matching*) use algorithms that require an IED template to be extracted from known IED images. The IED images have to be segmented to remove the background before the template can be generated.

1.1.3 The Difficulty of Cutting IEDs from a Photograph

The main part of this thesis is evaluating existing algorithms for image segmentation and create new algorithms to meet the special requirements of separating IEDs and their background. Figure 1.1 shows some colored images [5] typically used to evaluate the quality of segmentation algorithms.

Note that all objects shown in Figure 1.1 are completely visible and colored differently than the background. The edges are connected and define a closed surface. In contrast, Figure 1.2 shows real-life examples of IEDs. The biggest difference is that IEDs are not easy to detect even for humans. They are purposefully hard to detect, concealed or colored very similar to the background. Therefore it is not clear how well existing algorithms perform, if they are used to cut IEDs.

Figure 1.1: Examples of images similar to the images which were used to test the performance of GrabCut. The examples are part of the *Berkeley Segmentation Data Set and Benchmarks 500*. Typically the object is of a different color than the background and not concealed.

Figure 1.2: Examples of IEDs provided by TRADOC. The IEDs in the second row appear to be closed and monotonically colored. The boundaries are completely visible and in a good contrast to the background. The IEDs shown in the first row are partially hidden, so they do not appear as one closed surface.

1.2 Problem Terminology

1.2.1 Introducing Basic Terms

The following explained terms are used in this thesis.

- A *colored image* is the complete picture showing the IED and the surroundings before any processing on the image takes place. This image could be taken by a military patrol after detecting an IED.
- A *training image* is a colored image that shows the defined region of the IED, the acceptance region and the background. A typical training image is shown in Figure 1.3.

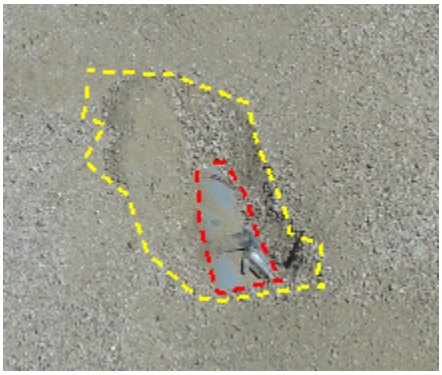

Figure 1.3: Example of a Training Image. The red polygon surrounds the region of the IED, and the yellow polygon surrounds the related acceptance region. All other pixels belong to the background of this image.

- A *generated training image* is a training image which was generated by combining an IED image and a background image. The source of a generated training image is not an actual colored image. As in a Training Image, in a generated Training Image the area of the IED, the acceptance region and the background are defined.

- *Image segmentation* is the process of partitioning a digital image into multiple segments. The goal of segmentation is to simplify and/or change the representation of an image into something more meaningful and easier to analyze [6]. Image segmentation is typically used to locate objects and boundaries in images.

- An *object* in image segmentation refers to a region in the colored image which is of particular interest. In general, this could be any kind of object such as a person, a flower, or an animal. In this thesis the object refers always to an IED.

- *Interactive image segmentation* allows the user to approximately pre-define the pixels which belong to the object or the background. Some of the algorithms we evaluate use *interactive image segmentation* to find a good balance between user interactions and achieved segmentation results. In the case of an object that is easy to separate from the background, the object pixels need to be only roughly defined to get a good segmentation result. If the result is not good enough, it can be improved by adding or excluding more object pixels. The amount of user interaction strongly depends on the algorithm used and the difficulty of the segmentation task.

- An *acceptance region* is the unusual region around the IED. This set of pixels of the colored image is not part of the IED but indicates that an IED might be present. Figure 1.3 shows such a region surrounded by a yellow polygon.

- A *background* is the the region which does not belong to either the IED or the acceptance region. Anything that is not "object" or "acceptance region" we call "background."
- *Labeling* in our work defines a process, where either the user manually or an algorithm automatically defines initial regions in the image.
- *Similarity* in the context of image segmentation measures the equality among different pixels. This measurement can be based on different criteria, i.e., structure, color, texture, etc.
- Opposite to similarity, *discontinuity* measure how different pixels are. In image segmentation, a high discontinuity among pixels usually means that they belong to different regions.

1.3 Research Objectives

1.3.1 Scope and Objectives

Our goal is to find a general approach for separating an IED from the background given a colored image. This objective must be achievable by soldiers without any special IT background by using their regular equipment. Another restriction of the problem is, that the segmentation has to be precise enough to allow generating a new image by combining the segmented IED and a different background.

Practically, our goal allows for the following sequence of actions:

1. A military patrol detects an IED and takes a picture of the IED and the scene.
2. An application developed in this thesis segments the image into IED and background.
3. The segmented image is either directly used as a training image in IED detection training, or to generate new training images.

All segmentation algorithms we found require object labels either as initial input or as iterative input to improve the result of the segmentation task. Because of the special circumstances while being on a military patrol and because IEDs are often partially hidden, it is generally not possible to provide object labels manually. Therefore, we design an algorithm that does not require the user to define the initial object pixels manually. Similar to the "GrabCut" [7], we

reduce the effort a user has to invest to achieve a good segmentation. But instead of decreasing the quantity of the user interactions, we provide a technique that results in more precise initial object labels.

The idea of our stepwise approach is to first generate a background model based on a user-defined background area. Based on this model, the algorithm highlights all unusual objects that do not fit the background model. The result of the second step is an overlay similar to the second image in Figure 1.4. The color codes indicate how well a particular pixel fits into the background model. The last step is extracting the initial object pixels. In an iterative process, the user provides a threshold for the gradient between blue and red. The set of all pixels above this threshold define the initial object label as shown in the third image in Figure 1.4.

To verify that our iteratively generated object pixels provide a better baseline for the actual segmentation than manually provided object pixels, we use both approaches as input for an existing segmentation algorithm and compare the segmentation results. For our purpose a segmentation result is considered as good if the segmented IED can be used to generate new training images.

We found a basic algorithm for image segmentation introduced by Boykov and Jolly 2001 [8] and several more advanced algorithms [7], [9], [10], [11] based on the same idea. But, to clearly separate the impact on the segmentation results achieved by using our estimated object pixels from the impact of using a more advanced algorithm, we use the basic algorithm introduced by Boykov and Jolly 2001 [8] for testing.

Out of scope:
Approaches discussed in the thesis try to estimate object pixels as a baseline for the actual image segmentation. The set of all estimated object pixels generates an initial object label. Once a background is defined, the initial object labels are fast to calculate and can be used to highlight "unusual" areas on an image as shown in Figure 1.4. Even if this result might be valuable for human operators to detect IEDs, detecting IEDs is not a direct objective for this thesis. In the context of this thesis, detecting unusual areas is a necessary step before segmenting an image.

This thesis is not creating new segmentation algorithms. Instead, we create algorithms to preprocess colored images in order to enable the usage of existing image segmentation algorithms for IEDs.

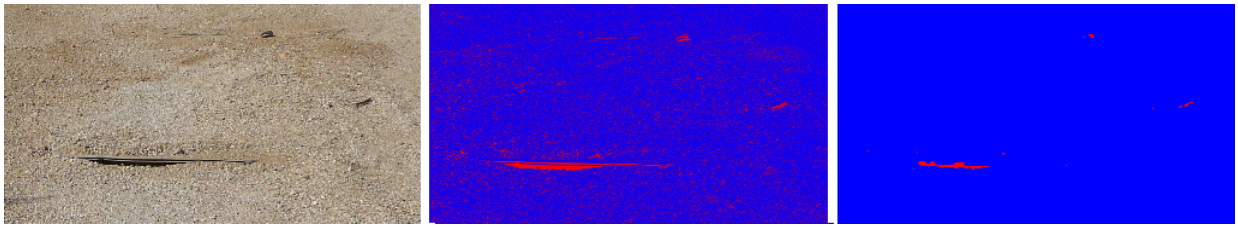

Figure 1.4: The left image shows a real-world example of an IED that is difficult to label. The image in the middle shows an overlay that highlights unusual areas on a image and the right image shows the estimated image pixels that are used to perform the segmentation.

1.3.2 Methodology

On an abstract level, the work described in this thesis can be divided in four main parts. Following a short explanation about these four working packages.

1. To verify that our implementation of the basic approach introduced by Boykov and Jolly 2001 [8] delivers qualitative similar results to other segmentation algorithms, we run a series of tests using colored images similar to the left image in Figure 1.5. This image does not show a realistic example of an IED detected by a military patrols. It rather shows an example of an object used to test the performance of segmentation algorithms in a non-military context.

Figure 1.5: The left image shows a colored image of an artillery-shell found in Iraq (image published by http://www.globalsecurity.org). The image on the right shows a real-world example of an IED detected by a military patrol.

2. We state the assumption that object labels cannot be provided manually for most of the detected IEDs. To test this assumption, we increase the grade of realism by applying the Boykov and Jolly algorithm to segment colored images showing real world examples of IEDs. The right image in Figure 1.5 shows an example of an IED in a realistic scenario. Since the background covers parts of the IED, it is not possible to draw a single area that includes all colors of the IED without including some background colors as well. Also, defining separate areas and combining them requires precision that is not present while being on a military patrol. Test series with a variety of manually provided object labels

and different realistic IEDs show that segmentation cannot be performed if the initial object pixels are not defined correctly.

3. Instead of asking a user to label both object and background prior to segmentation, our approach asks the user simply to provide a sample of the background. This background sample can even be provided using an entirely different image than the one intended for segmentation. Our algorithm then estimates the initial object label of the image intended for segmentation without further user input. This preprocessing enables the usage of existing image segmentation algorithms for IEDs. This part defines the main part of our thesis.

4. We use the basic segmentation algorithm of Boykov and Jolly 2001 [8] and run a series of tests using the same parameters as before, but with the automatically defined initial object pixels instead of the manually defined areas. The difference between the results of the test series using the manually provided object labels and the test series using the automatically generated object labels, is used to measure the goodness of our algorithm.

1.3.3 Benefit of the Study

Search-and-target-acquisition research benefits all soldiers. The ability to detect IEDs in complex scenes is a valuable skill, especially in today's operating environment. Determining ways to use colored images showing recently detected IEDs and generating new training images by combining IEDs and backgrounds expands the training-tool set for today's commander. Distributing this training across a variety of computer platforms promotes more accessible training for all soldiers, both in garrison and while deployed.

1.4 Organization of the Thesis

Chapter 2 provides an overview of approaches for image segmentation. Chapter 3 describes the family of algorithms for image segmentation we are interested in. In Chapter 4, we discuss why algorithms used for image segmentation cannot be directly applied to segment IEDs. Chapter 5 gives an overview of approaches to pre-process an image to achieve better segmentation results. In Chapter 6, we verify that our approach of estimating the initial object pixel results in a more precise segmentation. Examples how image segmentation is used to generate new training images for IED detection are provided in Chapter 7. Chapter 8 highlights the results and provides options how this work can be continued.

CHAPTER 2:
LITERATURE REVIEW

2.1 General Methods

The goal of image segmentation is to group pixels together into regions of similarity, based on various features extracted from an image. This might be color information that is used to create color models, or information about the pixels that indicate edges or boundaries or texture information [12].

The literature provides several criteria to group image segmentation algorithms. Pham, Xu, and Prince group the different algorithms based on the grade of user-interactions [13]. For our purpose, the grade of interaction is not of particular interest. We group the algorithms similar to Cufi, Munoz, Freixenet and Marti [14] based on the input information they require.

In general, image segmentation methods are based on two basic properties of the pixels in relation to their local neighborhood: *discontinuity* and *similarity* [14]. Figure 2.1 gives an overview of algorithms, categorized by the information they require to segment a colored image into object and background regions.

The following list indicates the main idea of the algorithms shown in Figure 2.1.

- Model-based methods segment an image based on existing geometrical or statistical models of the object itself. Since IED types quickly change, models of particular types are often not available. Also, to generate a template of an IED, we need to segment a similar IED from another image. This approach is not well suited in the context of IEDs and we exclude this type of algorithms from our review.

  However, we do use the idea of building a statistical model. Instead of building a model for the object, we build statistical models for the background. The detailed explanation of our statistical model is shown in Chapter 5, *Novel Segmentation Algorithm Development*.

- Boundary-based methods rely on discontinuities in image values between distinct regions, and the goal of these segmentation algorithm is to accurately demarcate the boundary separating these regions [15]. An example of an boundary-oriented approach is *Live wire*.

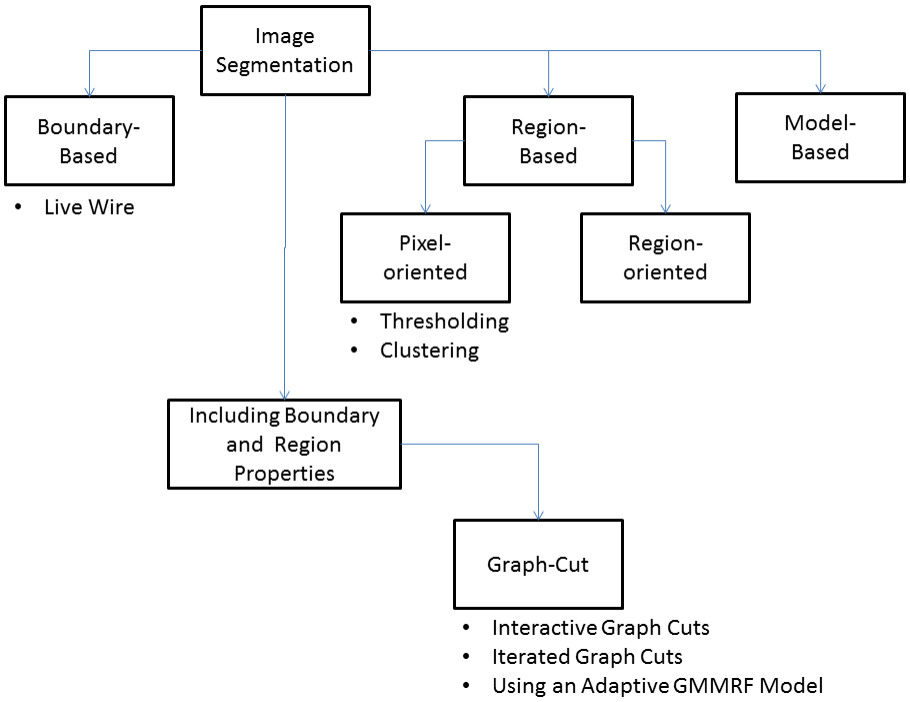

Figure 2.1: Algorithms for image segmentation categorized by the information they are including to segment a colored image into object and background regions.

- Region-based methods are based on the similarity property of the pixels belonging to the same area. The goal of these segmentation algorithm is to partition regions according to common image properties. Examples of image properties are *intensity values*, *textures* or *patterns*. Region-based methods can be sub-divided into *pixel-oriented* and *region-oriented* algorithms. Commonly used examples for pixel-oriented algorithms are *Thresholding* and *Clustering*.

- Algorithms combining both boundary-based and region-based methods aim to improve the segmentation result by including both the discontinuity and the similarity property. Solely boundary-based and region based methods often fail to produce accurate segmentation results, although the location in which each of these methods fail may not be identical [16]. Both approaches suffer from a lack of information since they rely on ill-defined hard thresholds, which may lead to wrong decisions. The goal of combined approaches is using the complementary information of region-based and boundary-based information to reduce the problems that arise in each individual methods [15].

Following is an overview and short explanation of all algorithms that influence our work.

2.1.1 Livewire

Livewire is also known as *Intelligent Scissors*. The basic idea of this algorithm is to formulate a colored image as a directed weighted graph. The pixels of the colored image represent nodes in the graph, connected with each of the eight adjacent pixels. The costs of the edges represent the discontinuity between their represented pixels. Different implementations of this algorithm use different criteria to describe the abstract term *discontinuity*. But, in general, we can say that similar adjacent pixels have a low discontinuity and therefore result in small costs associated with the corresponding edges.

Assuming that the colored image is transferred into a directed graph, we can find the path that minimizes the costs between two given pixels on the image. This path is the *shortest path* in terms of costs. If a user provides a sequence of pixels $z_0, z_1, .., z_{n-1}, z_n$ that are close to the object's border, the algorithm calculates the shortest path for each pair of consecutive pixels (z_{i-1}, z_i) for all i in $[1, n]$. Each of these shortest paths becomes a line segment, and the set of all line segments defines a boundary that segments the object from the background. It is the user's responsibility to select a sequence of pixels that results in a good segmentation.

There exists several different implementations of this algorithm [17], [18], [19], [20], [21]. They mainly differ in the way the user can add new points to the boundary and the cost calculation representing the *discontinuity* between adjacent pixels. Since we are not focused on a algorithms that include boundary information only, we do not review the different input possibilities.

Boundary-based algorithms put a lot of effort into calculating the discontinuity between adjacent pixels. These calculations are of interest for our study as well. All approaches we review do include the *Gradient magnitude*, the *Gradient direction* and *zero crossings of the second derivative* into their cost calculation [17], [22]. All of these operators are used for edge detection. In our approach, we include only the gradient magnitude as a measurement of similarity in color between adjacent pixels. The detailed calculations are shown in Chapter 5, *Novel Segmentation Algorithm Development*.

2.1.2 Thresholding

Pixel-based methods are the simplest segmentation technique and define the segmentation based on individual pixels without their neighborhoods. The input to a thresholding operation is typically a gray scale or color image. Thresholding is based on the assumption that different regions in an image have distinct intensity or color distributions as well. Based on this assumption, the

regions of an image can be segmented based on the mean and the standard deviation of each distribution. In simple implementations for gray-scale images, the segmentation is determined by a single parameter known as the intensity threshold. Each pixel in the image is compared with this threshold. Based on the comparison, each pixel is individually categorized as "belonging to the object" or "belonging to the background."

Figure 2.2 shows an example of a two-dimensional gray scale image and the corresponding intensity histogram. Assuming the intensity thresholds T_1 and T_2 are known and the intensity of a pixel at position (x, y) is $I(x, y)$, the object can be segmented from the background by applying following compound threshold rule: If $I(x, y) \geq T_1$ and $I(x, y) \leq T_2$ than (x, y) gets categorized as "belonging to the object."

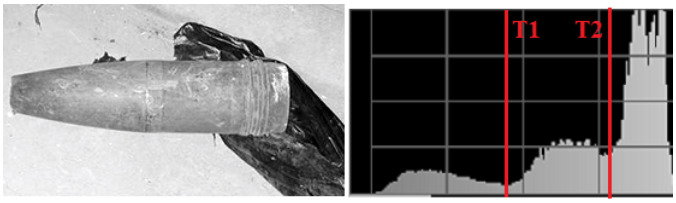

Figure 2.2: The first row shows a gray-scale image that can be segmented into object and background by applying a compound threshold rule. The second row shows the segmented IED and the associated intensity histogram.

For color images, different thresholds for each color channel or specified cuboid in RGB space can be used to replace the single parameter. In any case, the key parameter in the thresholding process is selecting the threshold value. Figure 2.3 scenario A shows an example where defining good threshold values is possible but difficult since the IED and the background show similar colors and therefore the RGB histograms overlap partially. In the case of scenario B it is impossible to assign threshold values in RGB space due the completely overlapping RGB histograms.

IEDs are often colored similar to the background, and the actual IEDs are only a small fraction of the complete image. Basing a segmentation only on the color of a pixel, and not including any neighborhood information, does not seem to be the right approach in the context of image segmentation for IEDs. However, we use the RGB histograms or the cuboid in RGB space to estimate the probability that the color of a specific pixel tends to belong to the background or to the object. The calculations we use are explained in detail in Chapter 5, *Novel Segmentation Algorithm Development*.

For a more detailed overview and explanation regarding the different approaches, see the survey by Sahoo, Soltani and Wong in 1988 [23]. Also, Sezgin and Sankur 2004 and Shapiro 2001

Figure 2.3: Comparing the usefulness of histograms to define the threshold for separating IEDs from the background. The image in scenario A shows an IED where thresholding based on the RGB histograms could be used to separate the IED and the background. Scenario B shows a real-life example of an IED where the color histograms completely overlap. For scenario B, thresholding in RGB space cannot be used to segment the object.

categorize the different techniques based on the information the algorithm manipulates [24] and based on finding a threshold [25].

2.1.3 Clustering

In general, clustering is an unsupervised learning task aiming to partition a set of data into clusters, such that the inner class similarity and the heterogeneity between the different clusters is maximized. For image segmentation we like to partition an image into k clusters, s.t. all pixels belonging to the same cluster are similar, and pixels belonging to different clusters are different. Depending on the purpose of the segmentation, similarity can be defined based on *distance*, *color*, *connectivity*, *intensity*, etc.

Clustering algorithms can be classified as *hard clustering* and *fuzzy clustering*. In the case of hard clustering, every pixel belongs to one and only one cluster, where for fuzzy clustering a decimal number in $[0, 1]$ indicates the tendency to belong to a cluster [26].

Hard Clustering

A well-known hard clustering algorithm is the *K-means algorithm* first introduced by MacQueen in 1965 [27] and then improved by Hartigan and Wong [28]. Assuming that the number of resulting clusters is given as k, the algorithm works as follows:

1. Choose k pixels as initial centroids, one for each cluster.
2. Compare every pixel with every centroid and assign the pixels to the most similar cluster.
3. Re-calculate the centroids in RGB space for each cluster.

4. Repeat step 2 & 3 until the position of all centroids stay constant.

As a result, we get an image segmented into k clusters, and each cluster having it is own centroid. Taking only the pixel information into account, is not precise enough to segment objects colored similar to the background. But, it appears that clustering is a memory efficient method for representing color models. For our work, we use the K-means algorithm to generate a set of background models, explained in detail in Chapter 5, *Novel Segmentation Algorithm Development*.

2.2 Graph Cut Algorithms

Graph Cut algorithms state one approach to include the complementary information of region-based and boundary-based information into the segmentation. The general concept of using binary graph cut algorithms for object segmentation was first proposed and tested in Boykov and Jolly 2001 [8]. Since the segmentation optimized by graph cuts combines boundary regularization with region-based properties [29], this thesis is mainly focusing on image segmentation by graph-cut algorithms.

Image Segmentation using Graph Cut algorithms separates the segmentation into three different aspects of the problem:

1. The first problem is including prior knowledge into the segmentation process by defining the initial regions, background and object, shown on a colored image. This task becomes harder as distinguishing between foreground and background gets more difficult. Therefore, the particular colored image to segment has a great impact on the usefulness of a selected approach.

2. Once the initial regions are defined, an undirected graph representing the colored image and the defined regions has to be generated. All graph cut algorithms we analyze define the undirected network G(V,E) as a set of nodes V, edges E, and capacities C: $E \rightarrow \mathbb{R}$.

3. A minimum cut algorithm applied on the generated network is used to partition the network. Because of the way the network is constructed, the partitioning of the network is equivalent to the segmentation of the underlying colored image.

Even though all graph cut algorithms address these three aspects of the problem, the actual implementation of the different aspects, specially defining the initial regions, vary greatly and

are optimized for images within a particular domain. To provide an overview about the different efforts to provide prior knowledge to the segmentation, we explain three different labeling techniques in more detail.

2.3 Image Segmentation using Graph Cut

We found several algorithms for Interactive Image Segmentation. But even the more advanced implementations refer to the approach introduced by Boykov and Jolly 2001 [8]. Since this segmentation approach states the baseline for all more advanced algorithms, this algorithm is described first.

The Boykov and Jolly algorithm addresses the segmentation of a monochrome image. This algorithm divides an image into two segments, the object and the background. The user has to provide the following information:

1. A set of pixels belonging to the object.
2. A set of pixels belonging to the background.
3. A set of pixels belonging either to the object or the background.

Figure 2.4 shows how this is done. All pixels inside the red polygon definitely belong to the object. All pixels outside the yellow polygon definitely belong to the background. It is uncertain to which segment all pixels between the yellow and the red polygon belong. This representation is called *complete labeling*.

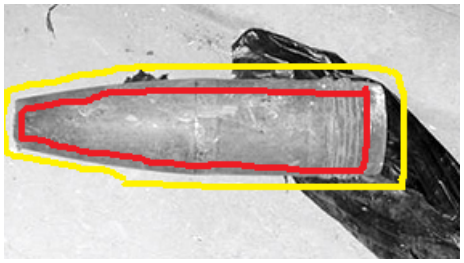

Figure 2.4: Monochrome image of an artillery-shell found in Iraq (image published by http://www.globalsecurity.org). The user defined the set of pixels which belong to the object or to the background.

In this approach segmentation depends on the generated object and background gray-level histograms. In Figure 2.5, the first row shows an monochrome image and the related gray-level histogram. The histogram shows that the monochrome image includes three different regions.

The second and third rows of Figure 2.5, show the image divided into object and background and the related histograms after segmentation. The gray-level histogram of the object shows one remaining region. This observation illustrates the assumption that for a good segmentation the opacity is coherent and the object appears to be closed.

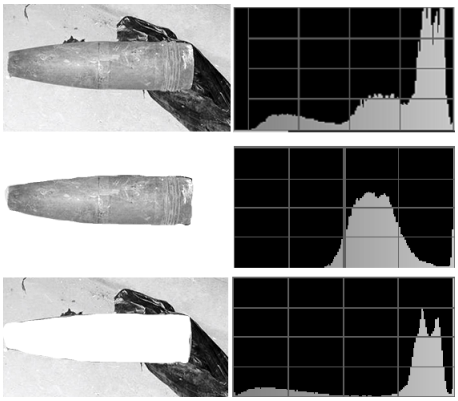

Figure 2.5: First Row shows a monochrome image of an artillery-shell found in Iraq (image published by http://www.globalsecurity.org) and the related histogram of the gray-value distribution. The second and the third rows show the histogram for the object and the background after they were segmented.

Chapter 3, *Minimum Cut Algorithm for Image Segmentation* explains in detail how the related undirected network is generated and how a Minimum Cut algorithm is used to segment the object.

2.4 Image Segmentation using "GrabCut"

This algorithm introduced by Rother, Kolmogorov and Blake 2004 [7] combines the following steps:

1. Obtaining a hard segmentation using an iterative Graph Cut [8]
2. Border Matting

Obtaining the hard segmentation

The main difference from the initial graph cut algorithm [8] is reduced user effort. Instead of defining the object and background region completely as shown in Figure 2.4, an example of complete labeling, this approach requires only an incomplete labeling as a first step. Initially only the pixels belonging to the background are defined as shown in Figure 2.6. The pixels belonging to the object are obtained automatically and not defined by the user. The segmentation

task follows the same principle as the Graph Cut. If the segmentation is not precise enough, the user provides additional labels to improve the result.Because every new label provides more information the segmentation task has to be repeated. The final result is derived in a number of iterations. Each of these iterations consist of a user input and a following segmentation. Therefore, this approach is called iterative Graph Cut [7].

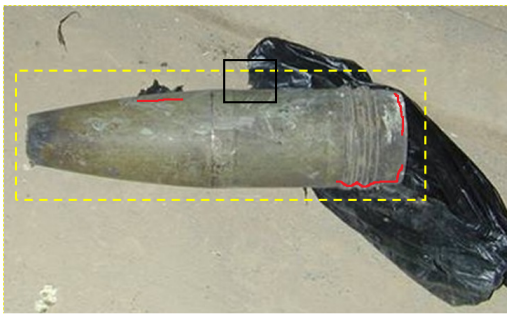

Figure 2.6: A colored image of an artillery-shell found in Iraq (image published by http://www.globalsecurity.org). The yellow rectangle shows the label the user added initially. Similar to the Graph Cut, every pixel outside this rectangular area belongs to the background. The red lines represent additional labels a user might have added in order to improve the result.

Border matting

In addition to the segmentation it might be necessary to produce a clean output of the object. However, mixed pixels along the object boundaries, belong to the object but their appearance changes depending on the background. The pixels close to the boundary between the object and a bright background appear to be different than the pixels close to the boundary between the object and a dark background. The occurrence of mixed pixels requires further image processing.

The GrabCut algorithm [7] uses border matting to deal with the problem of matting in the presence of blur and mixed pixels along smooth object boundaries. A simplified explanation of border matting is to color the mixed pixels similarly to the close object pixels. After adding transparency to these mixed pixels, such that the transparency increases as the distance to the border increases, the boundary appears to be smooth. For a more detailed explanation of border matting, see the “GrabCut” by Rother, Kolmogorov and Blake in 2004 [7].

2.5 Image Segmentation using an Adaptive Gaussian Mixture Markov Random Field (GMMRF) model

This approach introduced by authors Blake, Rother and Brown 2007 addresses the hard segmentation problem when the object and background color distributions overlap at least in parts.

One big difference from the two approaches explained above is the labeling. Instead of roughly estimating the object and the background separately as done for the basic graph cut algorithm, the regions are defined in one step by using a fat pen trail enclosing the object boundary. Figure 2.7 shows a fat pen trail enclosing the object boundary. Similar to the GrabCut algorithm, this approach allows adding more labels iteratively to improve the segmentation.

Figure 2.7: A fat pen trail encloses the object boundary. Every pixel outside the trail belongs to the background. Every pixel inside the trail belongs to the object. For all pixels covered by the trail, it is not clear to which segment they belong.

The regions of the object and the background define training regions to build a probabilistic formulation of the model in terms of a *Gaussian Mixture Markov Random Field (GMMRF)*. Secondly, a pseudo likelihood algorithm is derived to learn the colour mixture and coherence parameters for foreground and background. Li 1995 [30] and Perez 1998 [31] explain the different types of Markov Random Fields and their relation to (colored) images.

The segmentation task is to label all pixels in the "unclassified" region as belonging either to the object or to the background. Based on the following incomplete list of criteria and information, the algorithm decides if a pixel belongs to the object:

- Matching the labels of adjoining labeled pixels
- Models for color and texture properties of the object and background pixels. The models are learned from the related training regions.

Similar to the “GrabCut” [7], the user can add missing parts of the object by adding additional object labels.

A similar approach regarding the usage of Adaptive Gaussian Mixture Markov Random Fields is introduced by Kato and Pong 2006 [32].

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
MINIMUM CUT ALGORITHM FOR IMAGE SEGMENTATION

3.1 Detailed Example of the Initial Graph Cut Algorithm of Boykov and Jolly 2001

Graph Cut [8] of a monochrome image is a state-of-the-art algorithm for interactive image segmentation and the baseline for many more advanced algorithms. Since all algorithms we found require user-defined object labels, either as initial labels or iteratively added labels, we decided to start using the basic algorithm of Boykov and Jolly and improve it to match our requirements.

3.1.1 Input

The algorithm requires two different types of user input:

- The monochrome image showing the object and the background.
  This image is also expressed by a 1-dimensional array of gray values $\overrightarrow{z} = (z_1, .., z_n, .., z_N)$, where n is the number of pixels in a row, m is the number of rows and N is the total number of pixels, $N = n * m$.

- The initial estimation of object and background regions.
  The regions are user provided, as shown in Figure 3.1 image A. The separation of the image in the three regions – object, background and unknown – is called a trimap $T = \{T_O, T_B, T_U\}$, where T_O is the set of pixels labeled as object by the user, T_B is the set of background pixels, and T_U is the unknown region.

3.1.2 Optimization Variable

The segmentation of the image is expressed as a vector of unknown variables $\overrightarrow{\alpha} = (\alpha_1, .., \alpha_n, .., \alpha_N)$. The values of α_i show whether the related pixel tends to be a member of the foreground (object) or the background. Usually $0 \leq \alpha_i \leq 1, \forall i \in \mathbb{N}$.

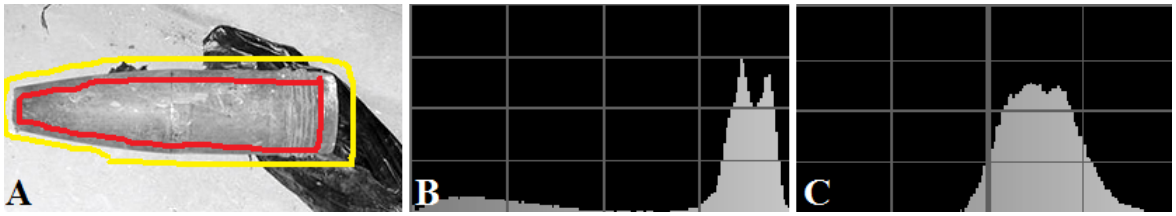

Figure 3.1: Image A shows an example of typical object and background labels used to define the training regions for the Boykov and Jolly algorithm of 2001. The Images B and C show the directly assembled gray-value histograms for the background (B) and the object (C)

A segmentation is a hard segmentation if $\alpha_i \in \{0, 1\}$. In this case $\alpha_i \equiv 1$ if the pixel z_i belongs to the object and $\alpha_i \equiv 0$ if the pixel z_i belongs to the background. We are focused on a hard segmentation.

3.1.3 Segmentation by Minimum Cut Algorithm

A pair of histograms is directly assembled from the user-defined training areas. The histogram in Figure 3.1 image B, shows the distribution of gray-values for all pixels belonging to the background. The second histogram shows the distribution of gray-values for all pixels belonging to the object. In this document, we call the histogram associated with the background $h(z, 0)$ and the histogram associated with the object $h(z, 1)$. Equation (3.1) shows the mathematical expression we use for the pair of histograms θ.

$$\theta \quad = \quad \{h(z, \alpha), \alpha \in \{0, 1\}\} \tag{3.1}$$

An objective function E, shown in Equation (3.2), is defined so that its minimum corresponds to a good segmentation. The literature calls this objective function *Gibbs Energy* or just *Energy* function.

For all Graph Cut algorithms we study, this objective function is the sum of the two functions U and V, where the function U is a region property which evaluates the consistency of the histograms of object and background, and the function V is a smoothness term/ boundary property which penalizes the difference between neighboring pixels.

For example, $U(\alpha, \theta, z)$ can be thought of a sum of functions, one for each pixel describing how the gray-value of pixel z_i fits into the assembled histogram of the object $h(z, 1)$. Similarly, $V(\alpha, z)$ can be thought of as a sum of functions on neighboring pixels with $V(z_i, \alpha_i, z_j, \alpha_j)$ measuring the similarity between adjacent pixels z_i and z_j. The function $V(z_i, \alpha_i, z_j, \alpha_j)$ results

in large values when similar pixels are separated and small values otherwise. If a pair of adjacent pixels is not separated, meaning their associated α values are equal, $V(z_i, \alpha_i, z_j, \alpha_j)$ is zero.

V is defined on a neighborhood of pixels. Depending on the neighborhood, the euclidean distance between adjacent pixels might be different, therefore V is a weighted sum using the $\frac{1}{dis(i,j)}$ as weights.

$$E(\alpha, \theta, z) \quad = \quad U(\alpha, \theta, z) + V(\alpha, z) \tag{3.2}$$

A good segmentation results in a small value for U because the gray-values of all pixels labeled as "belonging to the object" are consistent with the histogram $h(z, 1)$ and all pixels labeled as "belonging to the background" are consistent with the histogram $h(z, 0)$. A good segmentation also shows small penalty values for V. Figure 3.2 shows example values for both terms for one particular pixel z_i.

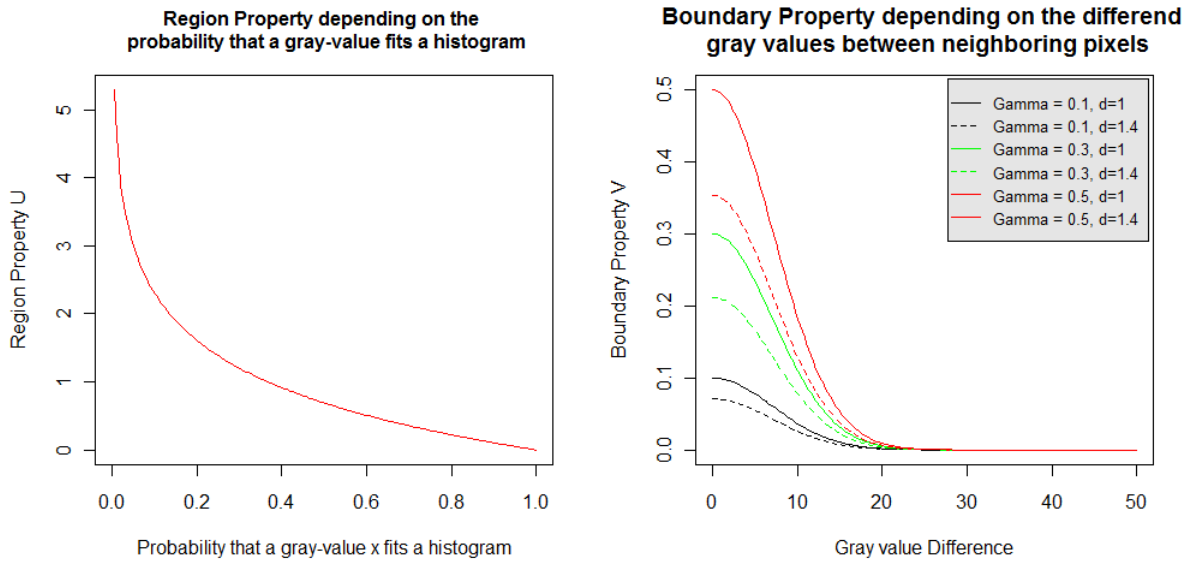

Figure 3.2: The plot on the left side shows the region property U depended on how the gray-value of pixel z_i fits into an assembled histogram. The plot on the right side shows the dependency between the boundary property an the difference in gray-value between adjacent pixels.

Since U and V are both penalty functions, a good segmentation can be obtained by finding the values $\overrightarrow{\alpha}$ that minimize the objective function E as shown in Equation (3.2). Meaning the smallest amount of *Energy* required to draw a border that segments the image into object and

background. Mathematically, we need to solve Equation (3.3) to find the segmentation for a given image.

$$\hat{\alpha} = \arg\min_{\alpha} E(\alpha, \theta, z) \tag{3.3}$$

3.1.4 A Segmentation Objective Function

We fully specify the objective function (3.3) using (3.4).

$$\begin{aligned} U(\alpha, \theta, z) &= \sum\nolimits_{n=1}^{N} -\log(h(z_n, \alpha_n)) \\ V(\alpha, z) &= \gamma \sum_{(m,n)\in C} \frac{1}{dis(m,n)} [\alpha_n \neq \alpha_m] e^{-\beta(z_m - z_n)^2} \end{aligned} \tag{3.4}$$

The expressions of (3.4), contain two unexplained constants γ and β. The constant γ is used to weight the smoothness term U compared to the region property V.

Boykov and Jolly 2001 [8] select β dependent on the expected difference in gray-value among adjacent pixels over an image sample as shown in Equation (3.5). Based on the results of Boykov and Jolly 2001, this choice of β ensures that the exponential term in V switches appropriately between high and low contrast. Other algorithms define β to be zero, to encourage smoothness everywhere on the image. If β is set to be zero, the resulting boundary property is called *Ising prior*.

$$\beta = \frac{1}{2E^2[z_m - z_n]} \tag{3.5}$$

3.1.5 Using the Minimum Cut Algorithm to Find a Hard Segmentation

Following, the segmentation algorithm defines an undirected network G(V,E) consisting of nodes V, edges E, and capacities C: $E \to \mathbb{R}$.

- Nodes V: All pixels z_i of the monochrome image become a node. In addition, two artificial nodes S and T are added. S represents the source or supply node, and T represents the sink or demand node. S and T are also called *terminals*.

- Edges E: The set of edges E consists of two types of undirected edges: neighborhood links and terminal links, or n-links and t-links for short. Each pixel z_i has two t-links (S,z_i) and (z_i,T) connecting it to each terminal. In addition, the n-link (z_i,z_j) exists if the pixel z_j is in the neighborhood of the pixel z_i.

  For image segmentation, the neighborhood is usually defined as a horizontal, vertical and diagonal neighborhood. This means that each pixel can have up to 8 neighbors.

- Capacities C: The capacities for edges between pixel-nodes correspond to the smoothness term in Equation (3.4). The different algorithms use different formulas for creating these terms. Table 3.1 gives the capacities of the edges similar to Boykov and Funka-Lea 2006 [29]. Different to Equations (3.5) and (3.4), Boykov and Funka-Lea 2006 use the parameter σ to refer to the expected difference in gray-value among adjacent pixels over the image.

Edge	Capacity	for
(z_i, z_j)	$\frac{1}{dis(i,j)} e^{\frac{-(I_i - I_j)^2}{2\sigma^2}}$	n-links
(S, z_i)	inf. 0 $G(h(z,1), z_i)$	z_i belongs to object, $\alpha_i = 1$ z_i belongs to background,$\alpha_i = 0$ z_i belongs to unknown region
(z_i, T)	0 inf. $G(h(z,0), z_i)$	z_i belongs to object, $\alpha_i = 1$ z_i belongs to background,$\alpha_i = 0$ z_i belongs to unknown region

Table 3.1: Table gives the capacity of edges similar to the capacities given by Boykov and Funka-Lea (2006).

The segmentation of the image into object and background is achieved by finding the minimum s-t cut of the associated network G(V,E). An example of a completely defined network, the minimum s-t cut and the resulting segmentation is shown in Figure 3.3.

3.2 Modifying the Initial Graph Cut Algorithm to Segment Colored Images

The initial Graph Cut Algorithm, as explained in the first part of this chapter, addresses the segmentation of monochrome images. This section provides a rough overview about the necessary modifications on the initial algorithm to segment colored images as well.

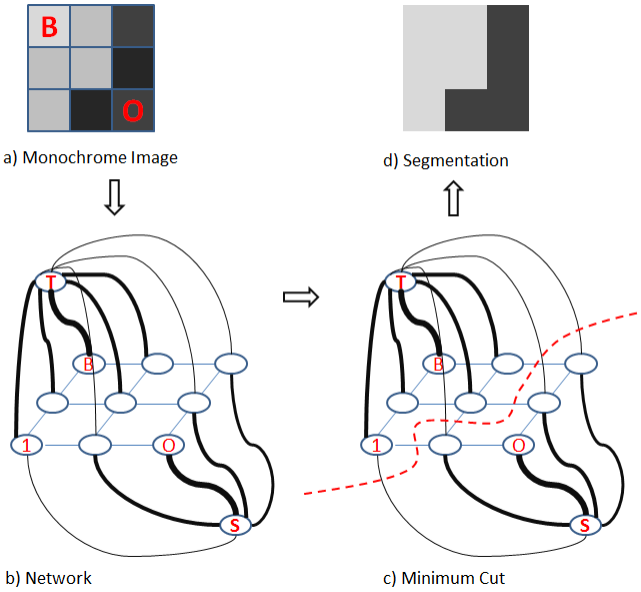

Figure 3.3: A segmentation example for a 3x3 monochrome image. a) Shows the three regions initially defined by the user. The pixel at position (3,3) belongs to the object, the pixel at position (1,1) belongs to the background, and all other pixels belong to the unknown region. b) Shows the network representation of the image. The capacity of each edge is reflected by the edge's thickness. c) Shows the minimum cut drawn as a red line. d) Shows the result of the segmentation task.

3.2.1 Defining the Initial Regions

For the first version of our algorithm, we follow the approach of Boykov and Jolly (2001) and ask the user to define both training regions, one for the object and one for the background, as shown in the two examples in Figure 3.4.

The two scenarios in Figure 3.4 show how the user-defined labels influence the segmentation algorithm and could change the results completely. This observation and potential solutions for this issue are discussed in detail in Chapter 4.

3.2.2 Generating the Object and Background Model

None of the Graph Cut algorithms we review uses the *RGB color model* since it is not sufficient for comparing color distributions [7], [9], [33]. Without questioning this statement, we first implement the naive approach and replace the initially used gray value histograms for the object and background by two sets of RGB histograms. Each set consists of 3 histograms, one for each color component in the RGB color space.

Figure 3.5 gives an example of a colored image, the user-provided object and background label and the resulting estimated color model for the object and the background. Note that the

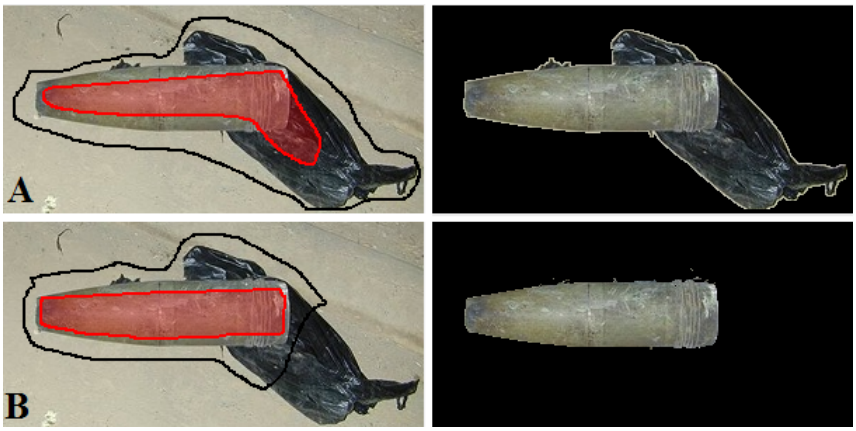

Figure 3.4: Two examples of object and background labels for the same colored image. In scenario A, the training area of the object includes the black bag and therefore segments the actual IED and the bag from the background. Wherein scenario B, the bag is included to the training area of the background and therefore only the IED is segmented from the background.

set of histograms also includes an overlay histogram. The overlay histogram shows a colored graphical distribution of all RGB colors where the tonal range for the RGB colors overlap. Both overlay histograms in Figure 3.5 show similar distributions among the related RGB histograms. Therefore the same segmentation result can be achieved by considering only one color component instead of all three. In the given example the histograms of the red component alone add enough information to segment the shown IED from the background. However, examples of real-world IEDs show that we do not have enough information to clearly decide if a pixel belongs to the IED or to the background.

3.2.3 Generating the Network Representing the Segmentation Problem

Besides the capacities of the edges. the undirected network G(V,E) is built the same way as for the segmentation of a monochrome image.

Different from the segmentation of a monochrome image, we include the information given by the three different distributions of the RGB color components. Table 3.2 gives the actual implemented capacities we use.

3.2.4 Finding the Segmentation – Nearly Orthogonal Latin Hypercube

The capacities of the t-links given by Table 3.2 use the parameter λ to weight the region and the boundary property. From a functional point of view, this parameter is similar to the parameter γ in the first section of this chapter.

The function we use to calculate the capacities of the n-links is similar to the function used

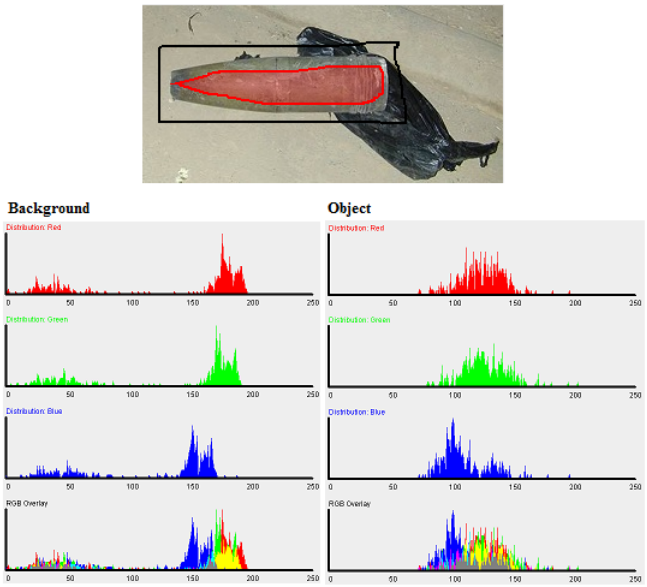

Figure 3.5: Image shows two sets of RGB color histograms. Each set consists of 3 histograms, one for each color component in the RGB color space. The fourth histogram is an overlay of the three separate RGB color components and shows a colored graphical distribution of all colors where the tonal range for the RGB colors overlap.

by Boykov and Jolly (2001). This function penalizes discontinuities between pixels of similar color when the average color difference is smaller than σ. The penalty gets smaller as the average color difference increases. Because of this behavior Boykov and Jolly (2001) refer to this parameter σ as *camera noise*.

Technically, we have a third "hidden" parameter indicating the number of decimals we use to define the edge capacities. There are some estimates available regarding the "best" values for λ and σ, but we do not know what number of decimals are sufficient. We run a series of test to explore the impact of these three parameters on the segmentation result.

To limit the number of tests to perform and to still have a space filling design of experiment, we use the *Nearly Orthogonal Latin Hypercube* [34], [35] design of experiment. An orthogonal design enables regression analyzes where it is not possible to actually perform a test for all possible combinations of all parameter values. We used an Excel spreadsheets published by MacCalman's 2012 to generate the design points for our series of tests.

The tests performed in this chapter have two purposes:

- To show that our implementation of Boykov and Jolly's algorithm can be used to segment simple IEDs from the background.

Edge	Capacity	Implementation
(z_i, z_j)	$V(z_i, z_j)$	$(1-\lambda)\exp^{-\frac{1}{2}(\frac{d}{\sigma})^2}\frac{1}{dis(i,j)}$ $d = \left(\sum_{c\in r,g,b}(z_{i,c}-z_{j,c})^2\right)^{\frac{1}{2}}$
(S, z_i)	$\begin{cases} inf & \text{if } z_i \text{ belongs to object} \\ 0 & \text{if } z_i \text{ belongs to backg.} \\ F(h(z,\alpha=1),z_i) & \text{otherwise} \end{cases}$	$\begin{cases} 8\max\{V(z_i,z_j)\}+1 \\ 0 \\ -\lambda\log P\{z_i \in \text{backg.}\|h(z,0)\} \end{cases}$
(z_i, T)	$\begin{cases} 0 & \text{if } z_i \text{ belongs to object} \\ inf & \text{if } z_i \text{ belongs to backg.} \\ F(h(z,\alpha=0),z_i) & \text{otherwise} \end{cases}$	$\begin{cases} 0 \\ 8\max\{V(z_i,z_j)\}+1 \\ -\lambda\log P\{z_i \in \text{object}\|h(z,1)\} \end{cases}$

Table 3.2: Table gives the first version of capacities we use to segment a colored image. The capacities are similar to the capacities shown in Table 3.1 but include the information given by the RGB color components.

- To define the domain for the parameters for the future tests.

For our first test series, we define the domain of the parameters as following: $\lambda \in [0, 1]$, $\sigma \in [20, 100]$ and the number of decimals in $[0, 4]$. We also manually define a "perfect" segmentation to compare the segmentation result for each design point. As measurement of goodness, we estimate the total *number of errors* by counting the misclassified pixels.

Figure 3.6 left plot shows that the number of errors varies a lot if the number of decimals is zero or one, but does not seem to get influenced as long as the number of decimals is at least two. Based on this observation, we keep the number of decimals constant equals to four for all future test series.

Figure 3.6 middle plot shows that there is a direct relation between the number of errors and λ indicating the weight of the region property. We get the highest number of errors, if λ is equals to one, meaning that only the region property is taken into account. This underlines our previous assumption that a segmentation using only the region property is not sufficient for our purposes. For all future test, we limit the domain of λ to $[0, 0.5]$.

This first test series does not provide any information that allows to limit the range of the parameter σ, as shown in Figure 3.6 right plot. Therefore, we keep to domain as $[20, 100]$ for the future test series.

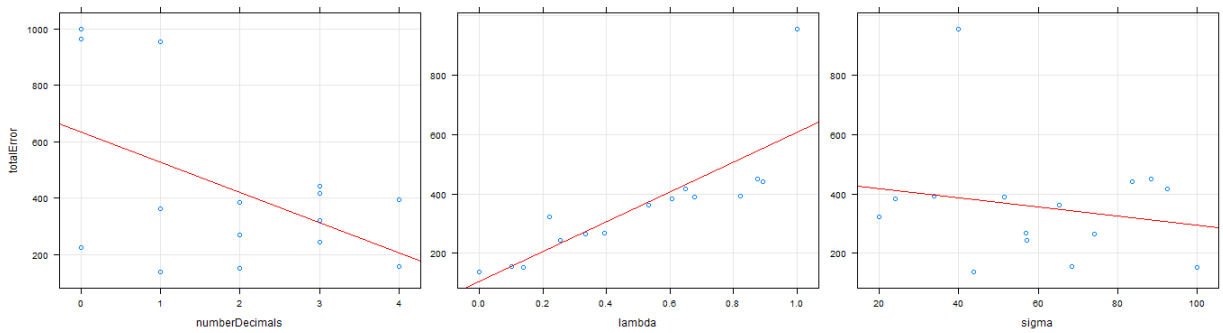

Figure 3.6: Results of first test series: Plots show the total number of errors dependent on the number of decimals (left), the values for parameter λ (middle) and the values for parameter σ (right).

Figure 3.7 shows two examples of simple IEDs used for the first test series, their training regions and the achieved segmentation results.

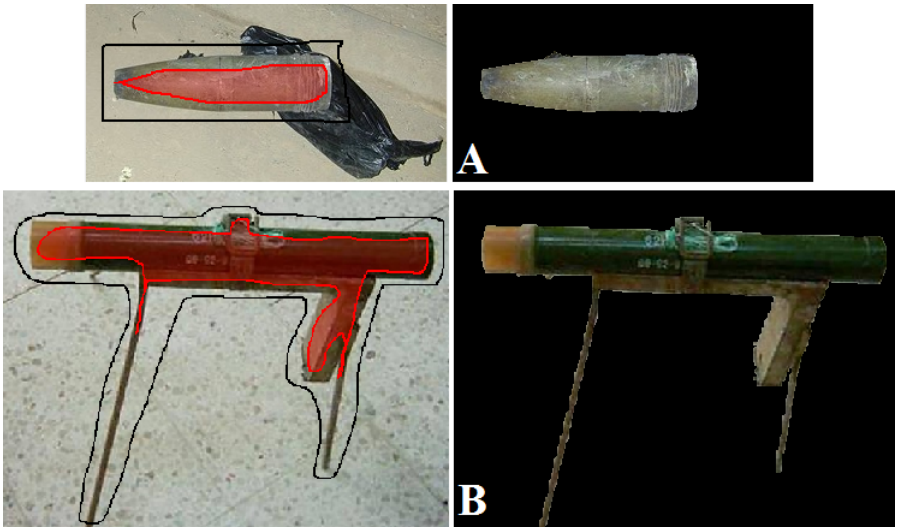

Figure 3.7: Two example of IEDs segmented from their background. The parameter values for both segmentations are $\alpha = 0.1$ and $\sigma = 100$.

CHAPTER 4:
FAILURE OF BASIC ALGORITHM

4.1 Example of a Real-World IED

Unfortunately, the implemented algorithm as shown in Chapter 3 cannot be used to segment a more realistic IED from the background. The image shown in Figure 4.1 left side, shows an IED that is colored similar to the background and also partially covered by material from the background. In order to generate a complete model for the object, it is necessary to include parts of the artillery shell and the cell phone into the training area for the object, as shown in Figure 4.1 middle image. The image in Figure 4.1 right side shows that using the same parameter settings as of Chapter 4 is not sufficient for good segmentation results.

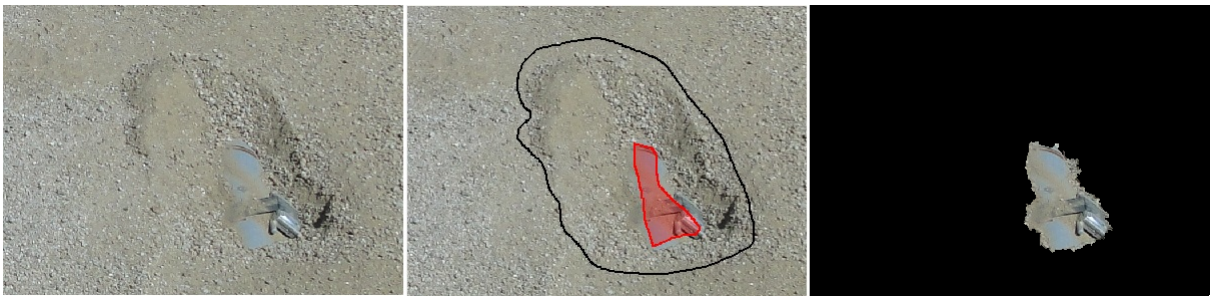

Figure 4.1: Example of a real-world IED that is more difficult to segment from the background.

4.2 Weaknesses of the Basic Algorithm

We identify two major issues that are due the different type of objects we like to segment from the background. Figure 4.2 shows a complete colored image that needs to be segmented.

Different from other segmentation tasks, real-world IEDs are only a small fraction of the complete image. Meaning that the training area for the background model is extremely large compare to the training area of the IED. Not surprisingly, as bigger the training area of the background gets, as more different color colors codes are included to the model of the background. And therefore, while building the network representing the segmentation problem we overestimate the capacities for the edges connecting the *pixel nodes* and the *T-node*.

Another weakness is that IEDs do not necessary appear as a single object. Often, charges and initiation systems appear -if visible- as different objects. Figure 4.2 shows an IED that consists

of two different parts. These different parts cannot be included into one single training area without adding background as well. In other words, from the beginning we include background pixels to the final result since we labeled them as "belonging to the object." Wrong initial labels cannot result in a good segmentation.

Another issue is due to a wrong assumption related to the color models we build. Replacing the single histogram of gray-values by three independent histograms, one for each color component assumes independence between the color components. But, we know that the RGB color components are not independent. Therefore, we can say that the color model we use for Chapter 4 is not precise enough, even if it is sufficient for simple IEDs.

The development of different potential solutions in order to solve these three issues is shown in Chapter 5, *Novel Segmentation Algorithm Development*.

Figure 4.2: Example of a complete colored image we need to segment. The shown IED consists of two parts at different locations. Therefore the IED does not appear as one single object.

CHAPTER 5:
NOVEL SEGMENTATION ALGORITHM DEVELOPMENT

5.1 Defining the Training Area for the Background

This section explains approaches we use to define the training area for the background. The biggest difference from images used for the algorithms listed in Chapter 2 is the small fraction of pixels belonging to the object. An easy approach to handle this issue is to manually define a subimage that only shows the IED and the close surroundings. This workaround avoids the problem instead of solving it. Therefore, we aim for an approach that takes the colored image as it is and builds a model considering the complete background area.

5.1.1 Single Training Area for the Background

Figure 5.1 gives a typical example of a colored image we need to segment. The image consists of 3456*2592 pixels where less than 1% do belong to the actual IED. As a result of this imbalance between IED and background area, the RGB color histograms for the background include almost the complete color space of the actual image.

Based on our assumption, the training area for the IED cannot be provided manually. Therefore, we like to estimate the IED area by identifying colors unlikely to appear in the user-provided background. But, since the background includes almost the complete color space of the image, no color codes are unlikely.

5.1.2 Multiple Training Areas for the Background

The main idea for this approach is based on the our answer for the question: *What makes a color unusual?* Considering the image shown in Figure 5.1, the color of the actual IED is not unusual. The same color is observed frequently in the rocky area of the image. But, in the background the colors of the IED are never observed on the street. Or generally speaking, a color is unusual if the color is unlikely to appear together with its surrounding.

We cut the background into smaller rectangles of similar size, as shown in Figure 5.2, left image. As a result, we get several areas showing colors that appear next to each other in the

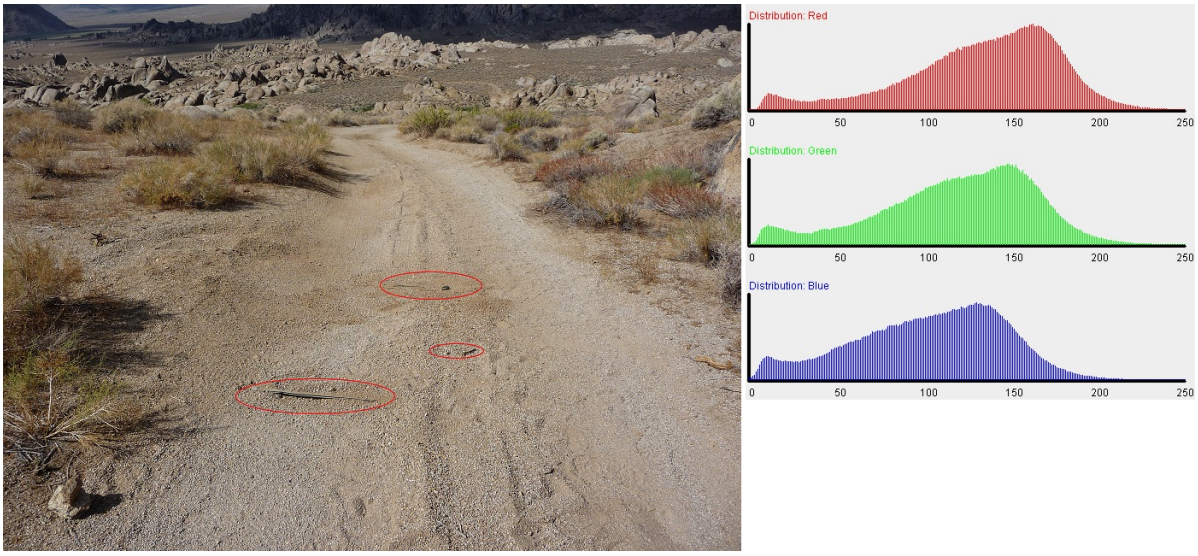

Figure 5.1: Example of a colored Image we need to segment and the resulting RGB histograms after using a single background area as defined by excluding the red circled areas.

background. This approach allows to define many color models of the background, without including all observed colors into a single color model.

It is theoretically possible to divide the background exactly between two different colors. In that case, none of the background areas would show the two different colors together, and the appearance of both colors would be classified as unusual. In all tests we do, this problem never appears. Nevertheless, Figure 5.2, right image shows artificially added overlapping areas to address this issue. The overlapping areas have similar size as the original rectangles and are centered at each corner of an original rectangle.

Figure 5.2: Examples of a user-provided training area of the background and the associated break down into smaller rectangles of the same size. In addition, the image on the right shows also the overlapping areas between adjacent rectangles.

5.2 Color Model Representing the Training Area of the Background

To evaluate the region property, we need to define a color model that describes the background. This measures how well a pixel's color fits into the background. This section explains the different color models we use.

5.2.1 Independent RGB Color Histograms

Boykov and Jolly's initial approach addresses the segmentation of a monochrome image, that can be fully described by a 1-dimensional vector of gray scale values. Different from gray scale values, RGB colors are 3-dimensional, and therefore cannot be expressed as a single 1-dimensional vector. Let RGB^* be the set of (r, g, b) tuples appearing in the background, than RGB^* is the color space of this background and a subset of RGB, a set that includes all possible tuples. Let r_i be the number of all tuples in RGB^* where the red color component is equals to i, and let i be in $[0, 255]$. We can think of r_i as it is defined in Equations (5.1), as the projection of RGB^* on the red color component. Similarly, RGB^* can be projected on the color components green and blue.

$$\begin{aligned} r_i &= |\{(r_i, g, b) \in RGB^*\}|, \forall i \in [0, 255] \\ g_i &= |\{(r, g_i, b) \in RGB^*\}|, \forall i \in [0, 255] \\ b_i &= |\{(r, g, b_i) \in RGB^*\}|, \forall i \in [0, 255] \end{aligned} \tag{5.1}$$

These projections can be visualized as three color histograms, one for each of the color components, as shown in Figure 5.1. After projection, the information regarding the occurrence of color codes is lost. To illustrate that estimating the region property based on projections is insufficient, we generate the three different images shown in Figure 5.3 left side. The images show different colors, and therefore have a different color space associated with them. But, the histograms of their projections are similar to the set of histograms shown in Figure 5.3.

Imagine, we use these as a color model to estimate if the RGB colors: *black* - (0,0,0), *red* - (255,0,0) and *violet* - (255,255,0) fit into the background. Then, the histograms let us believe that all colors *black*, *red* and *violet* are likely to appear in the background, which is wrong.

This example shows that projections of a color space are not sufficient to estimate the probability that a particular color code appears in the background, and can lead to wrong segmentation results.

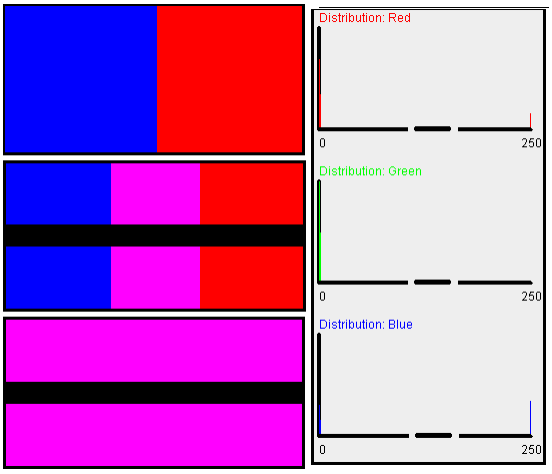

Figure 5.3: The images on the left shows three different backgrounds. All three backgrounds result in the same independent RGB color histograms, shown on the right side.

5.2.2 RGB Color Cube

Since the projection of the RGB color space on the different color components masquerades the information about the occurrence of color combinations, we use a 3-dimensional color model as symbolically illustrated in Figure 5.4. This *Color Cube* records the number of times each color code appears in the colored image. In this document, we use counter(r, g, b) to refer to the number of times a particular color code (r,g,b) occurs, and n_b to refer to the total number of pixels in the background.

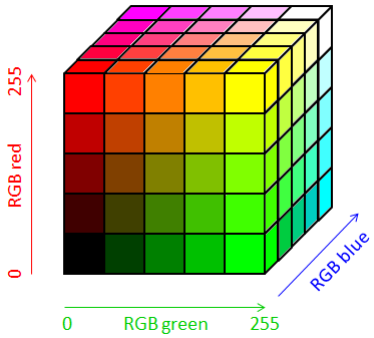

Figure 5.4: The image shows a simplified RGB color cube having a dimension of $5 * 5 * 5$. For our application, we use the complete color cube matching the complete RGB color space of 256*256*256 different colors.

The color cube allows an estimate of the probability that a particular color appears in the background by estimating the conditional probability that an observed pixel has this color, given that the pixel belongs to the background. The conditional probability can be estimated with a non-parametric estimator $\hat{p}_{(r,g,b)} = \frac{\text{counter}(r,g,b)}{n_b}$. This estimator has two major weaknesses.

The color cube contains about 16.8 million different colors, but the human eye is not able to distinguish all these colors. Therefore, a color that seems to occur frequently in an image, might be spread out among different color codes. Meaning, that the sum of these color codes is big, but each color codes does not occur often. For an example, imagine the complete image shown in Figure 5.1 is labeled as background. In this image, the maximum occurrence of a color $max(\text{counter}(r, g, b)) = 328$ while $n_b = 708,588$ and therefore the estimated probability that the color that appears most in the background is observed is only about $4.6 * 10^{-4}$.

The second weakness is that the color cube produces gaps between observed color combinations. For example counter(r, g, b) equals 0, while counter$(r - 1, g, b)$ and counter$(r + 1, g, b)$ are positive. In that case, the estimator for color code (r,g,b) evaluates to zero, but this is intuitively not correct. Figure 5.5, left side, shows the projections of one color cube on the three different color components. The differences in height between neighboring values of one color component are results of the gaps between neighboring pixels in the color cube.

5.2.3 RGB Color Cube Including Smoothing

We assume camera noise makes RGB colors be normally distributed, and use *Gaussian Curves* to fill possible caps in a generated color cube. In this document, we use the term *smoothing* to describe this process of filling the gaps. The main idea is to first define 3-dimensional Gaussians, each one centered at a color code that appeared in the background, and scaled by the number of times that particular color code appeared. The result of this first step, are several 3-dimensional Gaussians that might overlap in parts. In the second step, we build the 3-dimensional, cumulative distribution of all these Gaussian Curves.

Technically, we define one additional parameter σ that defines the *standard deviation* of all Gaussian Curves. Each Gaussian effects all color codes within a maximum euclidean distance of 3σ, how strong they are affected is evaluated by the height of the curve at this distance. Figure 5.5 shows the effect of smoothing using 3-dimensional Gaussian Curves with a σ equals three.

The smoothing process is computationally complex. Iterating through the color cube itself has

a complexity of n^3, where n is the number of levels for one color component. And for each of these steps, we need to iterate through all effected neighboring color codes that has another cubic complexity of $(2(3\sigma))^3$. Therefore, the overall complexity of smoothing is $6^3(n\sigma)^3$. For our color cube n is equals 256, and the σ is a user-defined parameter. For example, let σ be equals to three, then smoothing requires $97,84410^9$ steps.

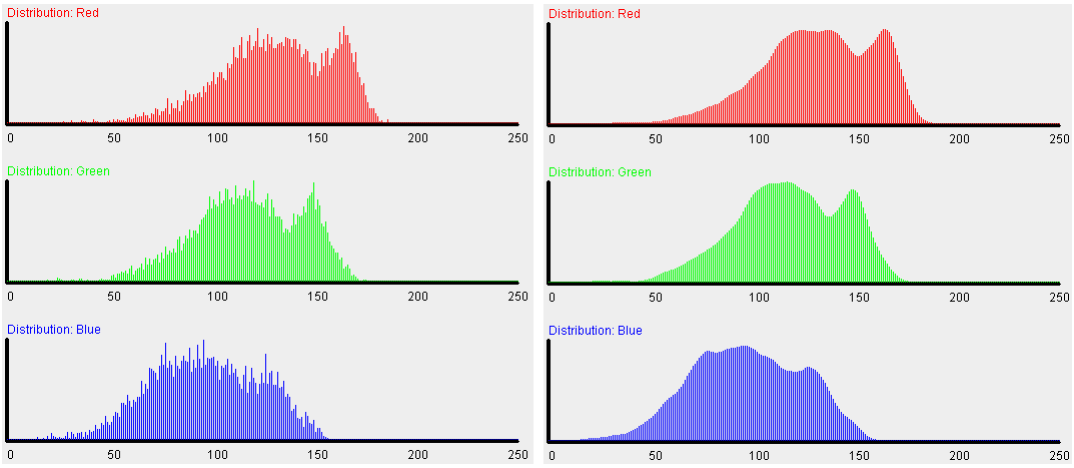

Figure 5.5: The image visualizes the effect of smoothing on the assembled RGB color distribution. For better visualization, these histograms show the projection of the same color cube on the red, green and blue dimension. Therefor, these histograms are not independent.

Smoothing solves the problem of gaps between neighboring color codes in the color cube, but it is computationally complex and requires a new parameter, the standard deviation defining the Gaussian Curves. Our main criticism of this approach is the user-defined standard deviation. Even if we get good segmentation results, the user may not know how to define this parameter.

5.2.4 Statistical Model Using Reference Points

The color cube is a sufficient model to describe the color distribution of an associated part of a colored image. But, the color cube has a dimension of 256^3, and each element in the color cube is in fact a variable to count the number of times a particular color code appears. To provide an example of the required working memory: In *Java* the variable type that requires the least memory and is sufficient in this context is *short integer* that requires 2 bytes. Therefore, a single color cube requires up to $256^3 * 2$ bytes, which is equivalent to 32,768 MByte.

In the previous section of this Chapter, we explain the advantage of cutting a single background area into smaller parts. In our experiments and depending the type of image, the number of background squares can be up to 200, or in terms of working memory 6,25 GByte.

Our *Statistical Model* is motivated by following key requirements:

- Using the same dimension as the RGB color space in order to not loose any information due to projections.
- Replacing the computational intensive smoothing without getting rid of the gaps-filling effect.
- Representing a color cube in a compressed way.

We use clustering to group all color codes present in the color cube. Different to the clustering explained in Chapter 2, where clustering is used to segment the image, we use clustering to find a compressed representation of the color cube. After applying the *k-means Clustering* algorithm for up to 10 clusters, a color cube is represented by the resulting clusters. Each cluster is defined by a *centroid* in RGB color space, the estimated mean color distance to a belonging pixel's color, the estimated standard deviation of this distance, and the number of pixels belonging to the cluster. Figure 5.6 shows an example of the set of centroids after applying *k-means clustering*. The complete statistics, as shown in Table 5.1 replaces one single color cube. Using this statistical representation instead of the original color cube, reduces the required working memory from initially 32,768 MByte to approximately 260 Bytes.

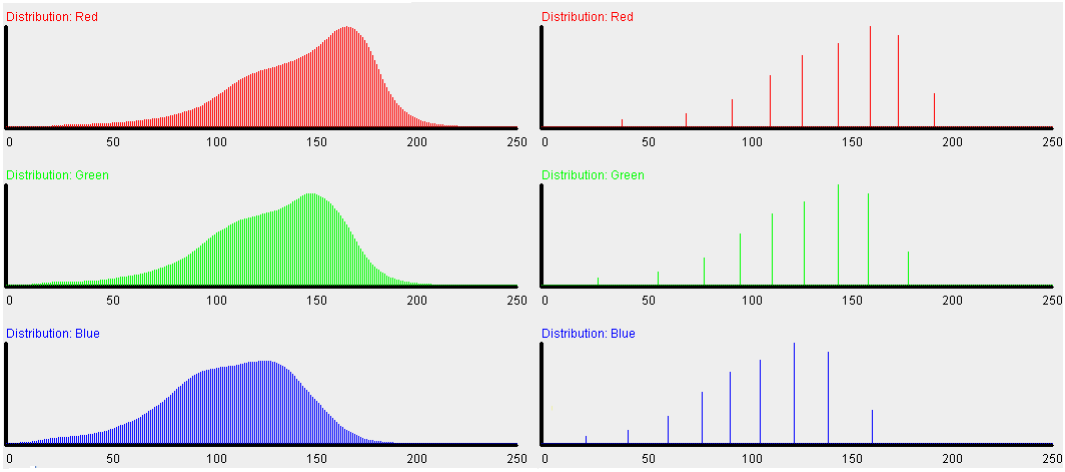

Figure 5.6: The set of histograms on the left shows the actual distribution of the RGB color components. The histograms on the right side show the centroids of the clusters after applying a *K mean algorithm* for $k \leq 10$. The reference point belong to the statistics shown in Table 5.1.

The information shown in Table 5.1 defines a 3-dimensional *Gaussian Mixture Model*. The projection of the mixture model defined by Table 5.1, on the red color component is shown in

Centroid in RGB color space	$\hat{\mu}_{distance}$	$\hat{\text{SD}}_{distance}$	Number of belonging Pixel
(178,163,143)	9.6	4.4	30274
(148,131,109)	10.2	4.2	27389
(95,81,63)	13.4	5.7	9116
(196,183,165)	13.9	10.6,	11051
(130,115,94)	10.9	4.7	23412
(72,58,43)	4.9	6.3	4598
(164,148,126)	9.4	3.9	32939
(114,99,80)	11.7	5.1	16912
(40,28,22)	15.3	7.8	2313

Table 5.1: Table provides an example statistics for the clusters after applying a *K mean algorithm* for $k \leq 10$. The centroids are also shown in Figure 5.6.

Figure 5.7, first row. The image shown in the second row of Figure 5.7 shows the cumulative curve that includes all Gaussian Curves defined by the statistics in Table 5.1.

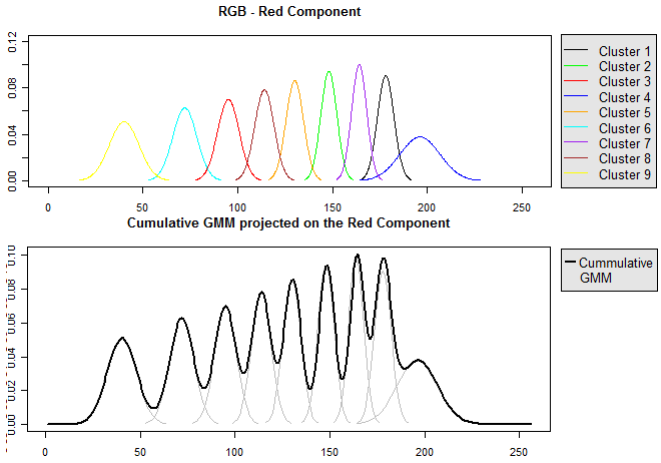

Figure 5.7: The figure in the first row shows the projection of the overlapping Gaussian Curves defined by the centroids and the standard deviations as shown in Table 5.1. The figure in the second row shows the cumulative curve including all Gaussian Curves of the figure in the first row.

Assuming that a cluster C_i is defined by a centroid (r_i, g_i, b_i), the estimated mean color distance $\hat{\mu}_i$ and the estimated standard deviation of this distance $\hat{\text{sd}}_i$. We use Equation (5.2) to calculate the probability that a color (r_1, g_1, b_1) belongs to this cluster.

$$P\{(r_1, g_1, b_1) \in C_i\} = 2\Phi\left(\frac{\sqrt{(r_1-r_i)^2+(g_1-g_i)^2+(b_1-b_i)^2}-\hat{\mu}_i}{\hat{\sigma}_i}\right) \tag{5.2}$$

$$P\{(r_1, g_1, b_1) \in B\} \leq 2\max_i\left(\Phi\left(\frac{\sqrt{(r_1-r_i)^2+(g_1-g_i)^2+(b_1-b_i)^2}-\hat{\mu}_i}{\hat{\sigma}_i}\right)\right) \tag{5.3}$$

For our purpose, we are interested in the probability that an observed color fits into a background B. But, a single background is represented by up to 10 different clusters. We compute an upper bound of the probability to observe a color (r_1, g_1, b_1) that belongs to one of these clusters using Equation (5.3).

5.3 Defining the Training Area for the IED

Different to the objects shown in Figure 1.1, IEDs can be concealed or camouflaged. In general, because of the nature of IEDs, the training area of IEDs cannot be defined by a single area, such that all colors present in the IED are included in the training region without including background colors as well. This section explains techniques we use to define the training area for the IED.

5.3.1 Single Object Label

Figure 5.8 illustrates the problem if we use a single object label to define the training area of an IED. In this example, it is impossible to define one single area that includes all colors of the actual IED without including background colors as well. The direct result of this insufficient object label is a wrong color model, similar to the set of histograms shown in Figure 5.8. Using this color model results in a segmentation that tends to include background at the boundaries of the actual IED. It is possible to reduce this effect by using only small weights for the region property for the segmentation, but this has more of a trial-and-error approach than an algorithm.

Using a single object label results in a segmentation similar to the example shown in Figure 5.8 at the top right. A new training image generated from this segmentation does not look right, because it includes background from the original image, as demonstrated in Figure 5.8 bottom right.

5.3.2 Multiple Object Labels

Figure 5.9 gives an example where we define multiple training areas, each one includes some colors present in the IED. Using multiple training regions, allows to include all colors present in

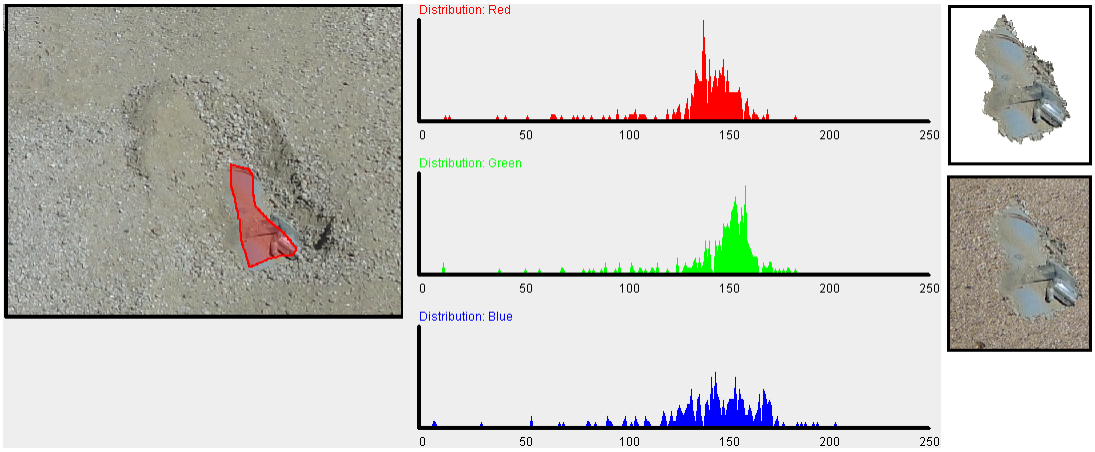

Figure 5.8: The image gives an example of a single object label and the associated set of RGB color histograms generated based on this user-defined training area. The image on the top right shows one segmentation result that illustrates the effect background color cause, if included into the color model of the IED. The image at the bottom right is the result of combining the segmented IED with a different background.

the IED without including areas that belong to the background. The color model for the IED is generated based on all training areas. The associated set of color histograms as shown in Figure 5.9 includes less background colors than the histograms shown in Figure 5.8. And therefore, the segmented IED includes less background than the segmented IED in the example of Figure 5.8. Since the background is completely excluded from the segmentation, the segmented IED can be combined with a different background, to generate a new training image as shown at the bottom right of Figure 5.9.

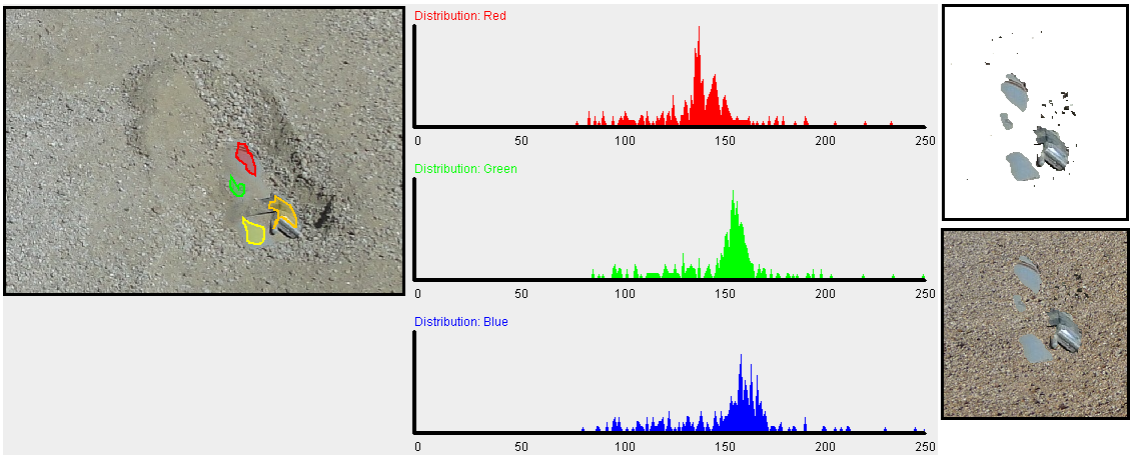

Figure 5.9: The image shows an example for using different object labels in order to include significant colors codes while excluding the color codes of the background covering the IED. Compare to the histograms shown at Figure 5.8, the joint object label include less background colors. The image on the top right shows one segmentation result. The image at the bottom right is the result of combining the segmented IED with a different background.

5.3.3 Estimated Initial Object Label

Multiple Object Labels work well, but in the context of IEDs this approach requires a precision that is for some IEDs close to segmenting the IED manually. Motivated by the idea of incomplete labeling, as it is used in Rother, Kolmogorov and Blake 2004 [7], we use the estimated distance to the background to estimate the initial object label automatically.

Assuming that a background B is represented by a set of clusters indexed by i and each cluster is defined by a centroid (r_i, g_i, b_i) and the number of belonging pixels w_i. Then, the weighted distance d of a pixel with color (r, g, b) to the background B can be expressed as shown in Equation (5.4).

$$\begin{aligned} d(r,g,b) &= \frac{1}{\sum_i w_i} \sum_i d_i(r,g,b) w_i \\ d_i(r,g,b) &= \sqrt{(r-r_i)^2 + (g-g_i)^2 + (b-b_i)^2} \end{aligned} \tag{5.4}$$

The maximum distance in the color cube we use is 442. Therefore, we build a one-dimensional vector with length 442 that stores a color-gradient from *blue* to *red*. To build a *heat map* showing the distance of each pixel color to the background B, we replace the actual pixel color by the gradient-color associated with the distance to B. Figure 5.10 left side shows an example of an IED we need to segment from the background. The image in the middle shows the heat map of the same image, where the actual colors are replaced with the color corresponding to the distance to the background B. The heat map visualizes the tendency that pixel belongs to the background. As more red a pixel is represented in the heat map, as more unlikely it is that the pixel belongs to the background. But, to generate a training area for the IED, tendencies are not enough. We need to define a criterion that decides, whether or not to include the pixel into the training area.

There are different possible solutions. We count the number of adjacent pixels that are closer to red than to blue. If this number is greater than half of the adjacent pixels, we include the pixel into the training area. Applying this criterion on the heat map results in an initial object label, as shown in Figure 5.10 right side.

5.4 Color Model Representing the Training Area of the IED

We use the same color models as we use for the background area. But, none of our examples of IEDs shows more than three colors that appear to be different for the human eye. Therefore, we

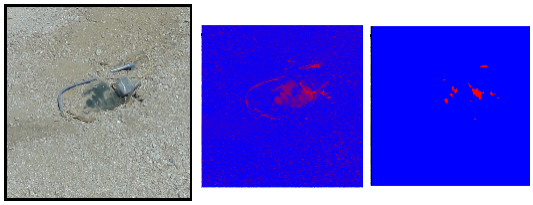

Figure 5.10: This set of images shows an example of an IED partially covered with material from the background. The image in the middle shows the estimated color distance to the background visualized by color radiant from blue to red. The area defined by the red pixels on the right image define the initial object label and provide the training area for the IED.

limit the number of cluster for statistical models representing the actual IED to three.

Similar to Equations (5.2) and (5.3), we estimate the upper bound of the probability that a color (r_1, g_1, b_1) belongs to the IED O as shown in Equation (5.5).

$$P\{(r_1, g_1, b_1) \in O\}\} \leq \quad 2 \max_i \left(\Phi \left(\frac{\sqrt{(r_1 - r_i)^2 + (g_1 - g_i)^2 + (b_1 - b_i)^2} - \hat{\mu}_i}{\hat{\sigma}_i} \right) \right) \tag{5.5}$$

CHAPTER 6:
EXPERIMENTAL EVALUATION OF NOVEL ALGORITHMS

6.1 A Complete Segmentation Algorithm

Imagine, we need to generate new Training Images showing the same IED type as in the colored image of Figure 6.1. To achieve this goal, we first have to segment the IED from the background. The segmentation has to be as precise as possible. But, we could tolerate missing minor parts of the IED. Whereas background included into the segmentation limits the further usage of the segmented IED.

Figure 6.1: The image shows the colored image used for the detailed segmentation example in Chapter 6.

6.1.1 Background Models

We first need to label the background training area as shown in Figure 6.2. The user defines the background area by drawing a rectangle, and the algorithm automatically divides this into smaller areas of similar size. The algorithm also adds overlapping areas, to ensure that all colors that appear close to each other, are covered by at least one area.

For each of these smaller areas, we generate a *color cube* that represents the complete color distribution. A *k-means Clustering* algorithm compresses the information into a statistical model. Let the background areas be indexed by n, and each background area b_n be statistically described by up to ten clusters indexed by i. The cluster $c_{n,i}$ refers to the ith cluster of the nth background area. Table 6.1 shows the set of clusters associated with the highlighted area in the background shown in Figure 6.2. The complete model *B* describing the user-defined background can be expressed as shown in Equation (6.1). The background model *B* might be stored for further segmentations showing a similar scenery.

$$\begin{aligned} B &= \{b_n | \text{ all small areas n}\} \\ b_n &= \{c_{n,i} | \text{ all clusters i in area n}\} \end{aligned} \tag{6.1}$$

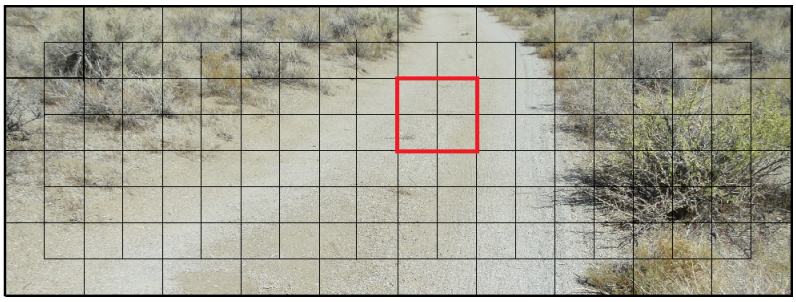

Figure 6.2: The image shows an example of a user-defined training area for the background. The big rectangle shows the user-defined area, whereas all smaller areas are automatically defined by the algorithm.

Centroid in RGB color space	$\hat{\mu}_{distance}$	$\hat{\text{SD}}_{distance}$	Number of belonging Pixel
(158,154,144)	16.65	12.25	121
(194,195,181)	5.65	2.56	5247
(204,205,193)	7.30	6.47	4613
(182,182,169)	7.83	4.01	1137

Table 6.1: Table shows the set of clusters associated with the highlighted area in the background shown in Figure 6.2.

6.1.2 Initial Object Labels

Figure 6.3 shows an example of the heat map of an unclassified region. Size and position of this unclassified region can be changed by the user. Every time the unclassified area changes, the algorithm calculates a statistical model with three clusters for the unclassified region.

The algorithms selects the background area that matches the unclassified area best. We call the background area that matches the unclassified model best, the *referenced background area*. Assuming, that the centroid of the ith cluster of the nth background area is given by $(r_{n,i}, g_{n,i}, b_{n,i})$ and the centroid of the jth cluster in the unclassified area is given by (r_j, g_j, b_j), then the *referenced background area* b_r is selected by minimizing Equation (6.2).

$$b_r = \arg\min_n \sum_i \sum_j \left(\sqrt{(r_{n,i} - r_j)^2 + (g_{n,i} - g_j)^2 + (b_{n,i} - b_j)^2} \right) w_i w_j \tag{6.2}$$

Once the referenced background area is computed, the heat map is generated by replacing the original color of each pixel in the unclassified area with the color that corresponds to the weighted distance to the referenced background area as shown in Equation (5.4). Figure 6.3 shows a heat map, where blue represents zero distance and red represents the maximum distance in the color cube.

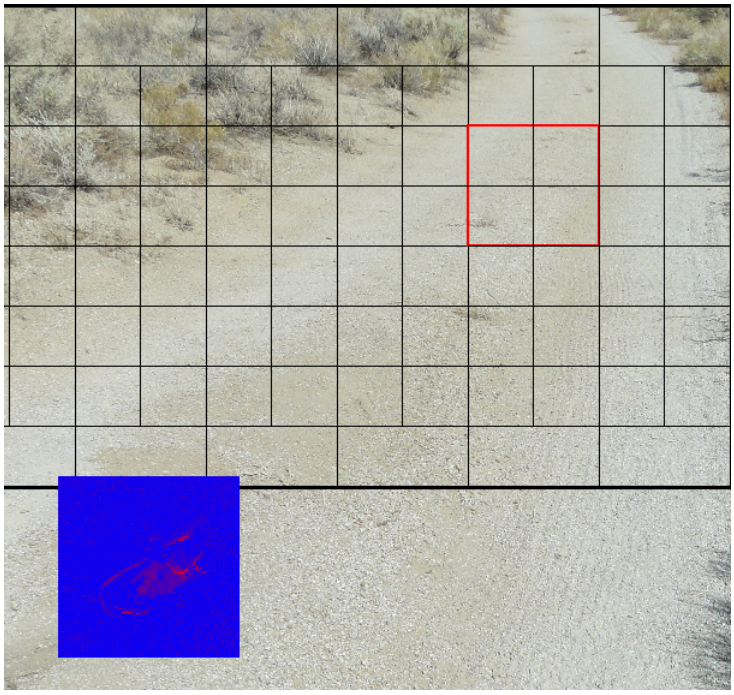

Figure 6.3: The image shows the heat map of an unclassified area. The heat map is generated based on the distance to the referenced background area highlighted in red. This background area is the best matching area for the current unclassified area.

A basic noise reduction, as explained in Chapter 5.3.3, performed on the generated heat map,

results in the initial object label as shown in Figure 6.4, right side. This estimated object label defines the training area for the IED. Meaning, that a new color model is built that includes all colors that correspond to this object label. As explained in Chapter 5, the statistical model of these colors is similar to the background models, but the number of clusters is limited to four.

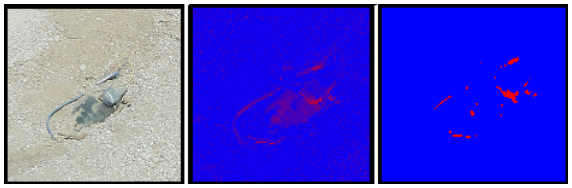

Figure 6.4: The image on the left shows the IED. The second image shows the same part of the image, but as heat map generated based on the referenced background area and the weighted distance of each pixel's color to this background area. The image on the right, shows the estimated object label used to perform the segmentation algorithm.

6.1.3 Constructing the Network

The subimage represented by the heat map as shown in Figure 6.3 is transformed into an undirected network G(V,E) as explained in Chapter 3.4.2. Table 6.2 gives the implemented capacities corresponding to Equations (5.3) and (5.5) used for the final version of our algorithm.

For the first test series, we vary σ and λ to explore the effect they have on the segmentation result. We compare the results of this first test series with a second test series where λ varies and σ is fixed and equals the expected color difference among adjacent pixels over the image.

Edge	Capacity	Implementation
(z_i, z_j)	$V(z_i, z_j)$	$(1-\lambda)\exp^{-\frac{1}{2}(\frac{d}{\sigma})^2}\frac{1}{dis(i,j)}$ $d = \left(\sum_{c\in r,g,b}(z_{i,c}-z_{j,c})^2\right)^{\frac{1}{2}}$
(S, z_i)	$\begin{cases} inf & \text{if } z_i \in \text{O} \\ 0 & \text{if } z_i \text{ belongs to backg.} \\ F(h(z,\alpha=1),z_i) & \text{otherwise} \end{cases}$	$\begin{cases} 8\max\{V(z_i,z_j)\}+1 \\ 0 \\ -\lambda \log P\{(r_i,g_i,b_i)\in B\}, \text{E(5.3)} \end{cases}$
(z_i, T)	$\begin{cases} 0 & \text{if } z_i \in \text{O} \\ inf & \text{if } z_i \text{ belongs to backg.} \\ F(h(z,\alpha=0),z_i) & \text{otherwise} \end{cases}$	$\begin{cases} 0 \\ 8\max\{V(z_i,z_j)\}+1 \\ -\lambda \log P\{(r_i,g_i,b_i)\in O\}, \text{E(5.5)} \end{cases}$

Table 6.2: Table gives the final version of capacities we use to segment an IED. The capacities are based on the equations shown in Chapter 5.

6.2 Results of the Segmentation Task

Similar to the initial parameter setting in Chapter 4, we define the domain of the parameters as: $\lambda \in [0, 1]$, $\sigma \in [20, 100]$ and the number of decimals equals to four. Due to the increased difficulty, it is impossible to define a "perfect" segmentation manually. Since we do not have a perfect segmentation we cannot use the number to mislabeled pixels as measurement of goodness. But, based on the assumption that a good segmentation results in closed areas, the number of isolated pixels can be used instead. The number of isolated pixels alone is not sufficient. For example, a segmentation that results in zero pixels labeled as IED is not a good segmentation, but the number of isolated pixels is zero. Therefore, we also evaluate the goodness of a segmentation visually and by combining the segmented IED with a new background.

We use the number of isolated pixels as a measurement of goodness. Since the term "isolated" is not well defined, for each pixel that belongs to the IED, we count the number of adjacent pixels belonging to the IED as well. Figure 6.5 visualizes the effect of removing the object pixels with less than n numbers of adjacent pixels belonging to an IED.

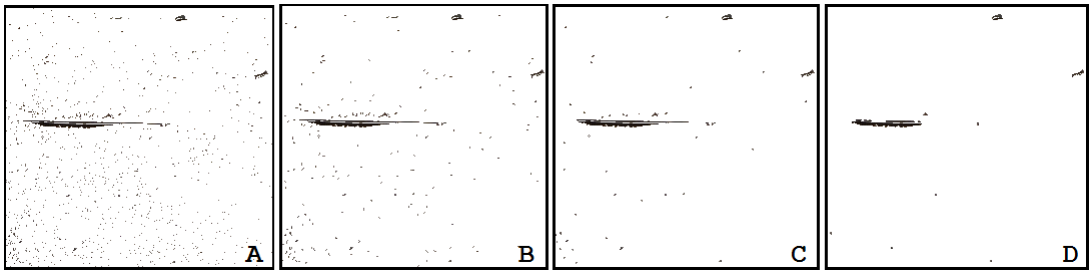

Figure 6.5: The series of images show the effect that noise reduction has on the segmentation result. Image A shows a segmentation result without any noise reduction. In image B, all object pixels with none adjacent object pixel are removed from the segmentation. In image B, C and D, all object pixels with less or equals to one, two and three adjacent object pixel are removed.

Examples similar to the one in Figure 6.5 show that the quality of the segmentation can always be increased by removing single object pixels and object pixels with only one other object pixel in their neighborhood. But, object pixels with more than one adjacent object pixels should not be removed.

Table 6.3 shows a part of a distribution for the number of adjacent object pixels. The relations between λ, σ and the number of adjacent object pixels are shown in Figure 6.6. Similar to the test series in Chapter 4, we can observe that as λ gets higher, the number of isolated object pixels increases. The effect of σ is opposite, as σ increases, greater differences in color code between adjacent pixels are accepted, which results in fewer isolated pixels.

λ	σ	$\lvert\{z_i \lvert z_i \in \text{IED}, n_i = 0\}\rvert$	$\lvert\{z_i \lvert z_i \in \text{IED}, n_i = 1\}\rvert$	$\lvert\{z_i \lvert z_i \in \text{IED}, n_i = 2\}\rvert$
0.88	88.57	875	297	75
0.25	57.14	304	220	52
0.67	51.42	732	314	79
0.89	83.60	908	303	83
0.53	65.31	385	231	62
0.10	68.57	212	178	46
0.33	74.28	296	202	45
0.13	100.00	164	148	32
0.22	20.00	710	374	93
0.82	33.88	973	382	99
0.64	92.51	378	215	55
0.60	24.11	912	388	101
0.39	56.85	341	235	61
1.00	40.00	1074	329	83

Table 6.3: Table gives the distribution for the number of object pixels in the neighborhood of each object pixel, for a segmentation that depends on the parameters λ and σ.

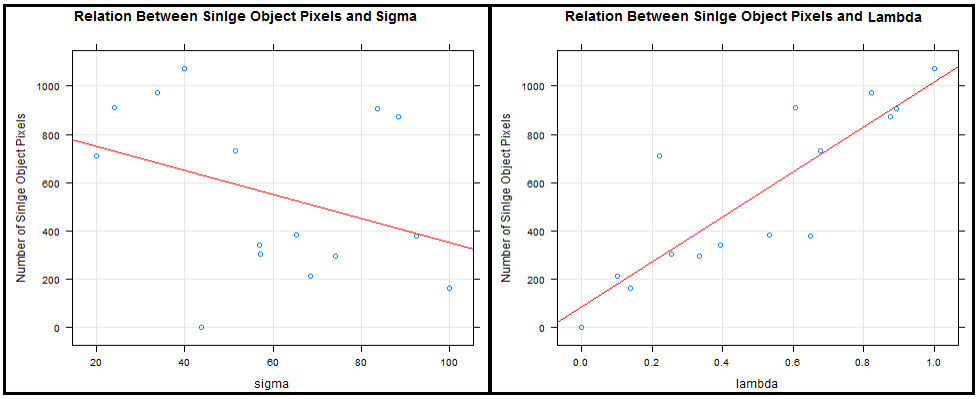

Figure 6.6: The plots show the general relation between the number of isolated pixels and the parameters λ and σ.

The effect of λ and σ are opposite, and the effect of σ itself is dependent on λ. Therefore, we fix σ, similar to Boykov and Jolly 2001 [8], to the expected color difference among all adjacent pixels over an image sample as shown in Equation (6.3). For the image shown in Figure 6.4, the expected color difference equals 46.65. The corresponding test series and associated results are shown in Table 6.4. The final test series limits the domain of λ to $[0, 0.5]$. As Figure 6.7 middle plot, shows, the range of isolated pixel is smaller if we fix the parameter β equals to 46.65, and decreases again, if we limit the range for λ. Therefore, all further test series depend only on the

parameter λ, and the range is limited to $[0, 0.5]$.

$$\sigma = E[z_m - z_n] \tag{6.3}$$

λ	σ	$\lvert \{z_i \lvert z_i \in \text{IED}, n_i = 0\} \rvert$	$\lvert \{z_i \lvert z_i \in \text{IED}, n_i = 1\} \rvert$
0.88	46.65	724	337
0.25	46.65	193	155
0.67	46.65	438	27
0.89	46.65	751	350
0.53	46.65	268	181
0.10	46.65	140	124
0.33	46.65	208	156
0.13	46.65	158	134
0.22	46.65	184	147
0.82	46.65	672	313
0.64	46.65	403	252
0.60	46.65	350	222
0.39	46.65	215	156
1.00	46.65	871	355

Table 6.4: Table gives the distribution for the number of object pixels in the neighborhood of each object pixel, for a segmentation that only depends on the parameters λ.

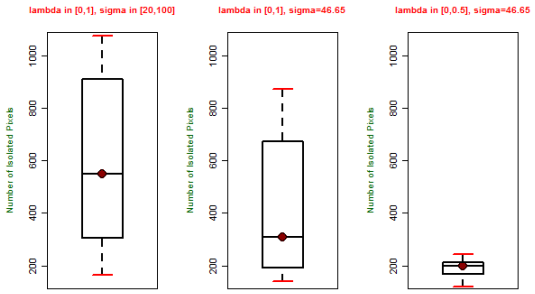

Figure 6.7: The plots show the distribution of the number of isolated pixels. The plot on the left shows the number of isolated pixels for λ in $[0, 1]$ and σ in $[20, 100]$. The plot in the middle shows the number of isolated pixels for λ in $[0, 1]$ and σ equals to 46.65. The plot in the middle shows the number of isolated pixels for λ in $[0, 0.5]$ and σ equals to 46.65.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 7:
POSSIBLE APPLICATIONS USING SEGMENTED IMAGES

The results of this thesis have three different applications: defining the IED on an image, generating new training images, helping humans detect IEDs. Following are some examples of these applications.

7.1 Defining the Area on an IED

To use a colored image for automated IED detection training, the area of the IED must be defined, in order to distinguish between a student's detection and misdetection. In some examples, the estimated object labels are close to the final segmentation result. In these cases the estimated object labels can be used to define the IED as shown in Figure 7.1. But, generally the estimated object label does not cover the complete IED as shown in Figure 7.2, and the area of the IED has to be defined based on the segmentation result.

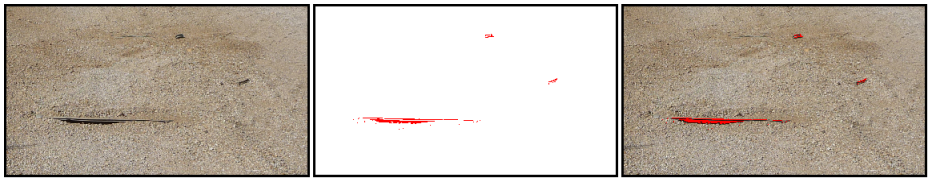

Figure 7.1: The image shows an unusual example where the estimated object label covers almost the complete IED and therefore defines the IED region for automated detection training sufficiently. In this case, performing the segmentation to define the IED area is not required.

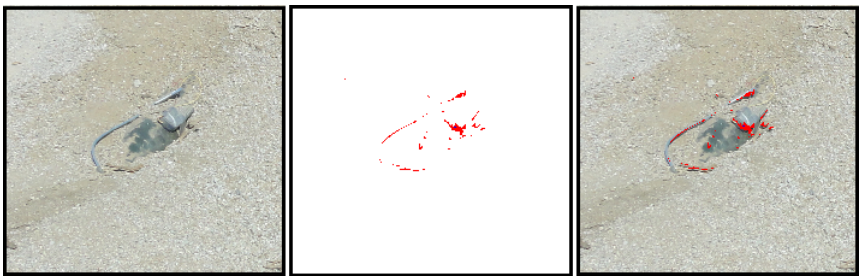

Figure 7.2: The image shows a usual example where the estimated object label does not cover the complete IED. In this case, performing the segmentation is necessary to define the IED area.

7.2 Generating new Training Images

To generate new training images, the IED has to be segmented as precisely as possible. In case of wrongly labeled pixels, we can tolerate object pixels that are labeled as "belonging to the background." But, background pixels that are labeled as "belonging to the IED" limit the further usage of the segmented IED. Figure 7.3 shows examples of a segmented IED combined with different backgrounds to generate new training images.

Figure 7.3: The image shows a segmented IED combined with different backgrounds to generate new training images.

Figure 7.4 shows the main problem with the current approach. The color of some IED parts fit well into at least one cluster of the background model. As a result these colors are not part of the estimated object label. As a result, object pixels with colors that fit the background model tend to be labeled as "belonging to the background." Figure 7.4 shows a segmentation scenario, where it would have been correct to include background colors into the color model of the IED. Chapter 8, *Conclusion and Future Work* discusses possible solutions to this problem.

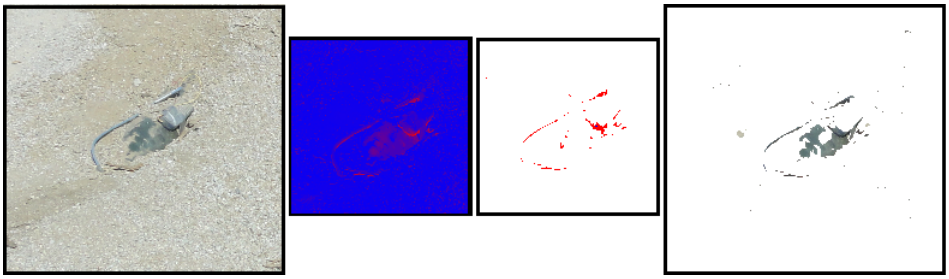

Figure 7.4: The image shows a segmented IED with a high rate of object pixels wrongly labeled as "belonging to the background."

7.3 Help detecting IEDs

Overlays replace each pixel color by a color code indicating the distance to the background. This functionality is not based on any assumption and can be computed at runtime. For all colored images showing an IED, the overlay highlighted either the IED or the unusual area around the IED. Therefore, overlays as shown in Figure 7.5 may be helpful as a hybrid between a computer pre-processing the image, and a human operator providing the final identification.

Figure 7.5 shows overlays for different parts of the image shown in Figure 5.1. Each row in Figure 7.5 shows the part of the original image we are interested in, the referenced background area and the corresponding overlay.

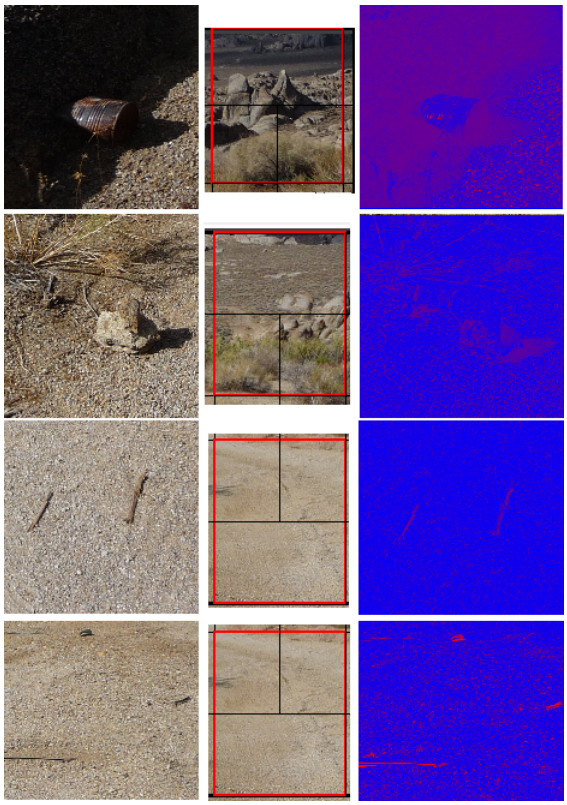

Figure 7.5: The image shows different parts of the colored image shown in Figure 5.1. Each row shows the area of interest, the referenced background and the generated overlay. Different from the first three images, the last image shows an IED. The visible parts of this IED are highlighted.

Our statistical background model can also be applied to images different than the image used to generate the model. Figure 7.6 and Figure 7.7 show examples of overlays based on a statistical model that is generated from a different image. Different to the IEDs shown in Figure 7.6, the IEDs in Figure 7.7 are more difficult to detect for humans. But, in both scenarios, the overlay hides the increased difficulty by highlights the IEDs. A possible application of statistical background models applied to different images is explained in Chapter 8, *Conclusion and Future Work*.

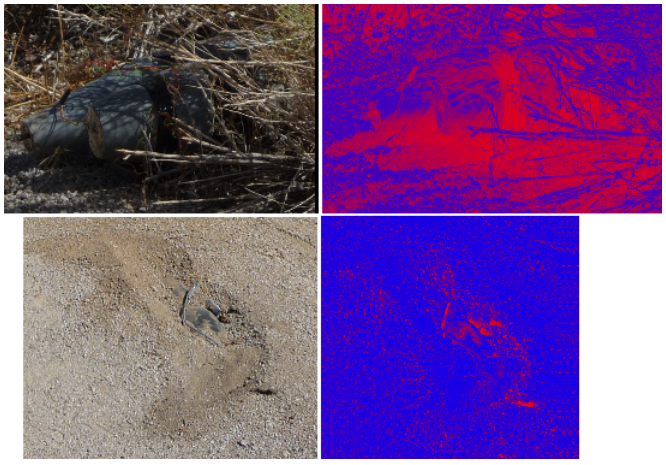

Figure 7.6: The image shows highlighted IEDs based on a statistical background model generated from a different image.

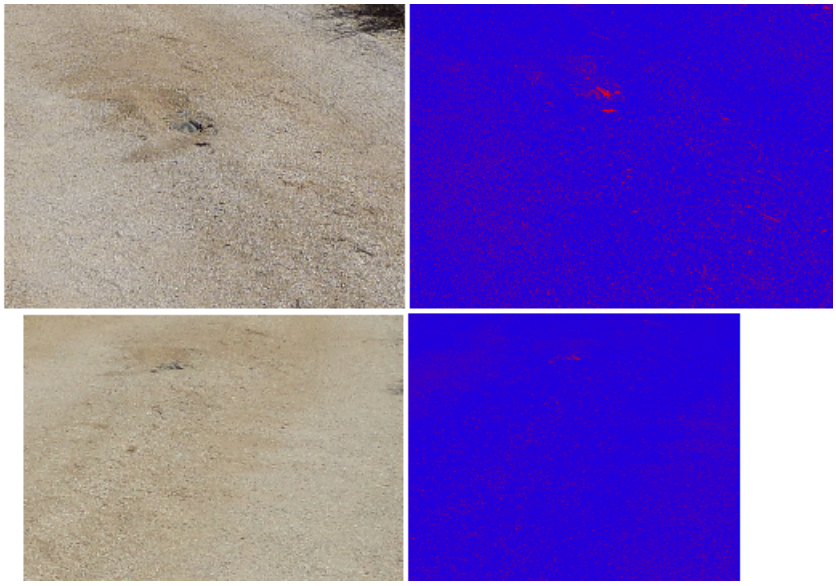

Figure 7.7: The image shows highlighted IEDs based on a statistical background model generated from a different image. Different to the IEDs shown in Figure 7.6, detecting the IEDs on the original image is more difficult.

CHAPTER 8:
CONCLUSION AND FUTURE WORK

We address the increased difficulty while segmenting IEDs from a colored image with three key contributions:

- Our algorithm automatically divides a user-defined background area into smaller areas. We generate separate color models for each of these area, to ensures that a color model includes only colors that appear in the same area of the background.
- We compress each of these complex color models into a statistical model, by applying a *K-means Clustering* algorithm. This increases the number of background models we can hold simultaneously in working memory, and allows to generate a set of background models that describes a complete environment.
- We estimate the initial object labels based on the color distance to the background. This approach enables generating color models for IEDs even if they are partially hidden or covered with background. It also allows IED segmentation without an user-provided object label.

Weak segmentation results as shown in Figure 7.4 are a result of an assumption made to estimate the initial object label. We assume that colors that are unusual for the background are part of the IED and include them into the initial object label. But, this also means that we do not include colors into the initial object label that are likely to appear in the background. This assumption is necessary for an unsupervised estimation of the initial object label. But as shown in Figure 7.4, this assumption is not generally true. The image shows, that a part of the IED, the display of the cell phone, is colored similar to the background, and therefore not included into the initial object label. This issue can be solved, by allowing the user to iteratively add missing parts and include them into the initial object label.

Currently, the region property is based on color information only. But, the dimension of the underlying model can be increased by adding more information like *infrared* or *temperature*. Using temperature based heat-maps is a common technique to detect IEDs that are completely hidden. Adding this information provides an additional criteria to classify a pixel as belonging to the IED or the background.

The final version of our algorithm is optimized for IEDs that are partially hidden, concealed and have a small area compare to the background. This optimized version results in weak segmentations if applied to IEDs that are easier to segment. Chapter 5 describes different approaches to label the initial object area. All of these approaches are helpful in some scenarios, and none of the approaches is generally better than the others. Therefore, combining all approaches to label the initial object area into a single application would generate a toolbox that supports the segmentation of many IEDs.

The statistical models we use to highlight unusual areas on an image, can be generated based on a different image. It is also possible to combine statistical models from more than one colored images. This could be helpful to describe the complete environment of an area and use an overlay that highlights all region on an image that do not fit into this environment. Before spending more time on overlay functionality, it should be tested, if the overlays have a statistically significant impact on the detection rate.

LIST OF REFERENCES

[1] A. H. Cordesman and J. Lemieux. IED metrics for afghanistan (Jan 2004 - May 2010). Presentation available from http://www.calameo.com/books/000009779cf4186fc669b. Accessed on 27. September 2012.

[2] L. O. Michael. Annual report 2010. Presentation available from https://www.jieddo.dod.mil/resources.aspx. Accessed on 27. September 2012.

[3] D. Naskrent. NATO air and space power in counter-IED operations. Presentation available from http://www.japcc.de/149.html. Accessed on 27. September 2012.

[4] C. Wilson. Improvised explosive devices (IEDs) in Iraq and Afghanistan: Effects and countermeasures. In *CRS Report for Congress*, p. 2. Congressional Research Service, 2007.

[5] D. R. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Technical report, EECS Department, University of California, Berkeley, 2001.

[6] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, Inc., Upper Saddle River, NJ, 2001.

[7] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, pp. 309–314. ACM, 2004.

[8] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. *Computer Vision*, 1(1):105–112, 2001.

[9] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *Computer Vision - ECCV 2004*, volume 3021, pp. 428–441. Springer Berlin / Heidelberg, Berlin, DE, 2004.

[10] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(1):167–181, 2004.

[11] S. Wang and J. M. Siskind. Image segmentation with ratio cut. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(6):675–690, 2003.

[12] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, Inc., Englewood Cliffs , NJ, 1982.

[13] D. L. Pham, C. Xu, and J. L. Prince. Current methods in medical image segmentation1. *Annual Review of Biomedical Engineering*, 2(1):315–337, 2000.

[14] X. Cuf, X. Muoz, J. Freixenet, and J. Mart. A review of image segmentation techniques integrating region and boundary information. In *Advances in Imaging and Electron Physics*, volume 120, pp. 1–39. Elsevier, 2003.

[15] X. Muoz, J. Freixenet, X. Cufi, and Marti.J. Strategies for image segmentation combining region and boundary information. *Pattern Recognition Letters*, 24(13):375–392, 2003.

[16] J. Freixenet, X. Muoz, D. Raba, J. Mart, and X. Cuf. Yet another survey on image segmentation: Region and boundary information integration. In *Computer Vision ECCV 2002*, volume 2352, pp. 21–25. Springer Berlin / Heidelberg, Berlin, DE, 2002.

[17] E. N. Mortensen and W. A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, 1998.

[18] E. N. Mortensen and W. A. Barrett. Interactive live-wire boundary extraction. *Medical Image Analysis*, 1(4):331–341, 1997.

[19] E. N. Mortensen and W. A. Barrett. Fast, accurate, and reproducible live-wire boundary extraction. In *Visualization in Biomedical Computing*, volume 1131, pp. 183–192. 1996.

[20] A. Chodorowski, U. Mattsson, M. Langille, and G. Hamarneh. Color lesion boundary detection using live wire. *Proceedings of SPIE Medical Imaging: Image Processing*, 5747(1):1589–1596, 2005.

[21] Z. Salah, J. Orman, and D. Bartz. Live-wire revisited. In *Bildverarbeitung fuer die Medizin 2005*, pp. 158–162. Springer Berlin Heidelberg, Berlin, DE, 2005.

[22] G. Hamarneh, J. Yang, C. McIntosh, and M. Langille. 3d live-wire-based semi-automatic segmentation of medical images. *Proceedings of SPIE Medical Imaging: Image Processing*, 5747: 1597–1603, 2005.

[23] P. K. Sahoo, S. Soltani, and A. K. Wong. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2):233–260, 1988.

[24] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004.

[25] R. Guo and S. M. Pandit. Automatic threshold selection based on histogram modes and a discriminant criterion. *Machine Vision and Applications*, 10(1):331–338, 1998.

[26] V. K. Dehariya, S. K. Shrivastava, and R. C. Jain. Clustering of image data set using k-means and fuzzy k-means algorithms. *Computational Intelligence and Communication Networks, International Conference on*, 0:386–391, 2010.

[27] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 281–297. University of California Press, Berkeley, CA, 1967.

[28] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[29] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *International Journal of Computer Vision*, 70(1):109–131, 2006.

[30] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, Inc., New York, NY, 1995.

[31] P. Perez. Markov random fields and images. *CWI Quarterly*, 11(4):413–437, 1998.

[32] Z. Kato and T. C. Pong. A markov random field image segmentation model for color textured images. *Image and Vision Computing*, 24(10):1103–1114, 2006.

[33] L. A. Consularo, R. M. Cesar, and I. Bloch. Structural image segmentation with interactive model generation. *Image Processing 2007 ICIP 2007 IEEE International Conference on*, 6(1): 45–48, 2007.

[34] T. M. Cioppa and T. W. Lucas. Efficient nearly orthogonal and space-filling latin hypercubes. *Technometrics*, 49(1):45–55, 2007.

[35] J. P. C. Kleijnen, S. M. Sanchez, T. W. Lucas, and T. M. Cioppa. State-of-the-art review: A users guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, 17(3):263–289, 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
USED SOFTWARE

Following applications are used while working on this thesis:

- Java SDK 1.6 is used to implement all versions of our algorithm. We use an implementation of Ford Folkerson's Minimum Cut algorithm that is developed by Robert Sedgewick and Kevin Wayne. We use a Java library, provided by MySQL, to connect our application to a MySQL database. Besides these external libraries, all required development were carried out by us.
- MySQL 5.2.3 is used to store the parameters for our test series and the associated results.
- R and the functionalities provided by the lattice package is used to visualize the relations between parameters and the measurements of goodness.
- An Excel worksheet provided by Alex D. MacCalman is used to generate the data points for a Second Order Nearly Orthogonal Design of Experiments.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California
3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California
4. Directory, Training and Education, MCCDC, Code C46
   Quantico, Virginia
5. Marine Corps Tactical System Support Activity (Attn: Operations Officer)
   Camp Pendleton, California
6. Director, Studies and Analysis Division, MCCDC, Code C45
   Quantico, Virginia